Karlsruhe Institute of Technology

KASTEL

Incentivized Privacy-Preserving Participatory Sensing

Masterarbeit

KIT – Karlsruher Institut für Technologie
ITI – Institut für Theoretische Informatik
Forschungsruppe Kryptographie und Sicherheit

Timm Lauser

18. Dezember 2019

Verantwortlicher Betreuer: Prof. Dr. Jörn Müller-Quade
Betreuender Mitarbeiter: Valerie Fetzer

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung des Karlsruher Instituts für Technologie zur Sicherung guter wissenschaftlicher Praxis in der gültigen Fassung beachtet habe.

Karlsruhe, den 18. Dezember 2019

(Timm Lauser)

Abstract

Participatory sensing systems utilize privately owned mobile devices and their embedded sensors to gather data. This is used in research projects in several fields such as environmental monitoring, public transport monitoring, and public safety. Due to the vast amount of information that can be inferred from sensor readings, such applications pose a serious risk to the privacy of the participants.

The Privacy-enhanced Participatory Sensing Infrastructure with Collusion Resistance (PEPSICo) model provides formal privacy and security guarantees for both, the participants contributing the data and the queriers that request and analyze it, but does not provide a way to reward participants for submitted data. However, as the participants incur costs through the sensing process, many systems will not be able to attract numerous participants without offering compensations. In this work, we extend PEPSICo with a mechanism to incentivize participants with rewards, without compromising the privacy of the participants. Therefore, we adapt Extended Black-box Accumulation (BBA+), a secure and private point accumulation scheme. Our proposed model provides strong security and privacy guarantees without requiring strong trust assumptions. Especially, transactions of honest users remain unlinkable even against malicious system operators colluding with the registration authority, queriers and other users. In opposite to PEPSICo, we do not have to trust the registration authority to behave honestly in this regard.

Zusammenfassung

Participatory Sensing Systeme machen sich die Vielzahl der Sensoren in heutigen Mobilgeräten zur Datenerhebung zunutze. Sie finden Anwendungen in Forschungsprojekten auf vielen verschiedenen Gebieten, z. B. in der Umweltüberwachung, der Beobachtung des öffentlichen Nahverkehrs und im Bereich der öffentlichen Sicherheit. Die Vielzahl der Informationen, welche von den erhobenen Daten abgeleitet werden kann führt jedoch dazu, dass solche Anwendungen die Privatsphäre ihrer Teilnehmer gefährden.

Das *Privacy-enhanced Participatory Sensing Infrastructure with Collusion Resistance* (PEPSICo) Model garantiert starke Datenschutz- und Sicherheitseigenschaften sowohl für die Teilnehmer welche die Daten liefern als auch für Datenverarbeiter. Allerdings bietet PEPSICo keine Möglichkeit, Teilnehmer für die beigesteuerten Daten zu entlohnen. Dies könnte viele Systeme davon abhalten, eine ausreichende Menge an Teilnehmern zu gewinnen, da diesen durch die Datenerhebung und -übermittlung auch Kosten entstehen. Diese Arbeit erweitert PEPSICo mit einem Anreizsystem, welches es erlaubt, Teilnehmer für die beigesteuerten Daten zu belohnen, ohne deren Privatsphäre zu gefährden. Dazu verwenden wir eine modifizierte Version von *Extended Black-box Accumulation* (BBA+), einem sicheren und Privatsphäre schützenden System zum Sammeln von Punkten, welche dann für Belohnungen eingetauscht werden können. Das vorgeschlagene Model bietet starke Sicherheitsgarantien und schützt die Privatsphäre der Teilnehmer, ohne dass auf das ehrliche Verhalten von Systemkomponenten vertraut werden muss. Insbesondere können Transaktionen selbst dann nicht als vom gleichen Benutzer ausgehend identifiziert werden, wenn böswillige Servicebetreiber mit der Registrierunsgstelle, Datenverarbeitern und weiteren Teilnehmern kollaborieren. Im Gegensatz zu PEPSICo gilt dies selbst dann, wenn sich die Registrierunsgstelle nicht an das Protokoll hält.

Contents

1 Introduction

The ubiquity of mobile devices such as smartphones provides lots of possibilities for data collection and analysis but also poses major challenges regarding user privacy. A participatory system uses a collection of such mobile nodes as its source of sensor data. These devices are controlled by their respective owners instead of a central observer, which is why the participation of a large group of people is required to utilize such a network.

However, collected information often includes Personally Identifying Information (PII). For example, if it contains location information and individual data reports are linkable, the system operator could misuse this information to create movement profiles of participating devices which, in the case of smartphones, often directly correlate to the movement of their owners. This poses a threat to the owner's privacy. While a lot of research has already been done in this field, many proposed solutions have strong trust assumptions and do not provide formal security and privacy proofs (cf. [Shi+11; Chr+13; GGP16; Con+19]).

An exception is Privacy-enhanced Participatory Sensing Infrastructure with Collusion Resistance (PEPSICo) [GMP14], which is a formal model addressing this issue by defining an architecture and protocols for a participatory sensing system. It effectively protects the user's privacy by preventing the linkage of data reports as originating from the same user. Moreover, it hides the submitted data and its context, the query identity, from the system operator and protects the privacy of the queriers that request and utilize this data by hiding the query identities of the reports they collect. However, it lacks a mechanism to amplify participation and attract more users. This is usually done with an incentive system, which allows compensating users for their efforts and computing time required to generate data reports by giving them incentive points that can be accumulated over many reports and at some point in time exchanged for goods or money.

A general model to implement such an incentive mechanism in a privacy-preserving way is Extended Black-box Accumulation (BBA+) [Har+17], which allows secure accumulation of incentive points without enabling the system operator to link transactions to individual users.

This work extends PEPSICo using BBA+ as a mechanism to provide incentives while preserving strong guarantees for the privacy of the participants. More precisely, we propose Incentivized Privacy-Preserving Participatory Sensing (I3PS), a general-purpose participatory sensing system with an incentive mechanism that provides the following features:

- Unlinkability of honest users during report submission, incentive collection, and incentive redemption, even if all other system entities collude. This property is not limited to honest-but-curious

system entities.

- Data reports are confidential towards the system operators and unauthorized queriers, especially, if they do not collude with a user registered for the same query, the query identity is hidden as well protecting the privacy of the queriers.
- Incentive rewards are bound to a user identity, thus, incentives cannot be shared between users.
- The number of reports an individual user is allowed to submit for a specific data query can be limited.
- The querier can verify that incentive rewards have been delivered correctly by the system.

The structure of this work is as follows: In Chapter 2, we discuss participatory sensing systems and their challenges in general before introducing cryptographic preliminaries in Chapter 3. Afterwards, we recapitulate PEPSICo (Chapter 4) and BBA+ (Chapter 5), as they are fundamental to our work. In Chapter 6, we introduce an interim model that combines these two models into a participatory sensing system with an incentive mechanism. However, this model has still some issues, wherefore, in Chapter 7 we introduce a second model addressing these issues.

2 Participatory sensing

To improve a given participatory system, it is important to understand the requirements of such systems. In this chapter, we first examine the meaning of the term *participatory sensing* and how it differs from similar and related terms. We then motivate the importance of participatory sensing by giving use case examples and discuss the main challenges for participatory sensing systems. Last, we give an overview of the existing research on privacy-preserving participatory sensing.

2.1 Definition

The term participatory sensing was introduced by Burke et al. [Bur+06] to describe a paradigm that focuses on the participation of users from the public sphere. Because of their ubiquity and high capabilities, mobile phones can be used to form interactive networks of sensors to collect data location- and time-aware. As the sensor devices are under the control of their respective owners instead of a central observer, the owner's participation is required. Moreover, the network itself is usually not owned by the operator of the participatory system and, therefore, potentially distrusted (cf. [HCG15]).

Goldman et al. [Gol+09] state three general approaches to participatory sensing:

Collective design and investigation
: The participatory system is collaboratively defined and used by a group of individuals. The community of individuals owns the entire process, being investigators and subjects at the same time.

Public contribution
: While the participants are actively involved in the collection of data, the definition of the research question and the use of the results is usually done by another individual or organization without their involvement. Recruiting interested individuals for the data gathering process allows acquiring data at a larger scale than it would be possible by conventional methods.

Personal use and reflection
: Within this approach, individuals record data about themselves for personal discovery, eg. to reveal hidden habits and patterns in their lives.

Within this work, we focus on the *public contribution* approach, which is the approach PEPSICo uses. Moreover, the other approaches are less critical regarding privacy as the data reported by the participants is not under the exclusive control of a third party. We only focus on the parts that are

important to user privacy, not on the complete participatory sensing process, mainly the transfer of the sensing data and the involved parties. However, the complete sensing process is described by Goldman et al. [Gol+09] as well.

There are three elementary roles in participatory sensing systems focusing on *public contribution* (cf. [HCG15]): The *participants/users*, that provide the data, the *queriers*, that request and utilize the data, and the *system operator*, providing the infrastructure and mediating between the participants and the queriers.

2.2 Similar and related terms

Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are widely used in industrial and environmental monitoring and are well researched. According to He, Chan, and Guizani [HCG15], the main differences to participatory sensing are that the sensor nodes, as well as the network, are under central control and ownership. They only sense the surrounding environment but typically not human behavior and, therefore, the main security concerns are the interception or modification of data in transit, alongside the disruption of routing packages and the malicious retasking of sensing nodes.

Mobile Crowd Sensing

Guo et al. [Guo+14] define Mobile Crowd Sensing (MCS) as "a new sensing paradigm that empowers ordinary citizens to contribute data sensed or generated from their mobile devices, aggregates and fuses the data in the cloud for crowd intelligence extraction and people-centric service delivery". In addition to participatory sensing, it reuses the user-contributed data from mobile internet services, such as mobile social networks. Therefore, it involves implicit and explicit participation. However, the term is also used as a synonym for participatory sensing (cf. [HCG15]).

Social sensing

The term *social sensing* was introduced by Wang, Abdelzaher, and Kaplan [WAK15]. Similar to MCS, it describes the combination of three data collection types:

Participatory sensing, where the user is actively involved in the data gathering process

Opportunistic sensing, where the user is only passively involved (eg. by pre-authorizing the gathering process)

Social data scavenging, where the user is unaware of the gathering process (eg. scanning social networks)

The main challenge is perceived to be the reliability of data and reported data is viewed as an unverified claim.

2.3 Motivation and examples

Utilizing participatory sensing can have several advantages over traditional WSNs [HCG15]. It does not require setting up a cost-intensive sensing infrastructure as it usually makes use of the large variety of sensors already included in mobile phones or wearable devices. Moreover, these devices also tend to have more computing resources available than traditional sensor nodes, which allows for more complex sensing applications. It is even possible to actively involve the participants in the sensing process. Participatory sensing systems are able to operate in environments that WSNs are not (which also applies the other way round) and are well-fit to collect space- and time-dependent data.

In 2012, Tilak [Til13] surveyed the real-world deployments of participatory sensing. At this time, they were primarily limited to small-scale research prototypes in the areas of health and fitness monitoring, environmental monitoring, transport and civil infrastructure monitoring, and urban sensing. However, these prototypes already wielded promising results and demonstrated the huge potential of participatory sensing applications. A recent survey from 2019 by Capponi et al. [Cap+19] shows that participatory sensing has indeed become quite successful and further areas of applications, such as emergency prevention, public safety, indoor localization, and waste management, have developed. They also provide a timeline of developments in participatory sensing and indicate its success factors.

In the following, we give a few examples of participatory sensing applications in some of the areas mentioned above.

Examples for environmental monitoring applications include GasMobile [Has+12], a low-cost system to monitor air pollution levels, and NoiseMap [Sch+11], which uses Global Positioning System (GPS) and microphone data to generate real-time noise maps.

In the area of public transport, Haig, Hayati, and Tomasic [HHT18] propose the usage of participatory sensing to detect crowded buses. They argue that the traditionally used Automated Passenger Counting (APC) systems are not always reliable and that transit information providers typically do not have access to this information in real-time. Their system uses the accelerators available in modern smartphones to predict the state of the riders (such as sitting, walking or running) with machine learning algorithms. In addition, Zhou, Zheng, and Li [ZZL12] predicts the arrival time of buses using sensing resources that impose a low power-consumption on participants' mobile phones, such as cell tower signals and movement statuses instead of GPS.

As an application in the field of public safety, iSafe [Bal+12] uses participatory sensing to predict the safety of users based on the crime statistics of their current location and the safety profiles of co-located users.

2.4 Main challenges

There are three main challenges in participatory sensing systems: User privacy, participant attraction, and data trustworthiness. User privacy strongly conflicts with data trustworthiness, as a high degree of user anonymity makes it very difficult to identify and eliminate users reporting false data (cf. [HCG15]).

User privacy

In participatory sensing systems, a lot of PII can be exposed to the system operator and the queriers. Apart from the actual sensor data, the data reports usually contain important contextual information such as location and time of the sensing. Moreover, multiple records can be linked together to profile users and derive further information. Additionally, there is the network and communication data, eg. the Internet Protocol (IP) address of the user and the frequency of reporting. This may also include the client and operating system used to communicate with the service. Lastly, observing which sensing tasks a user takes part in can reveal some information in itself.

For the actual sensor data, a privacy respecting system has to be transparent on exactly which data is collected and its possible relevance for privacy. This is especially important for systems where the user is not actively involved in the actual data gathering process, but where data is collected passively without requiring user interaction. Privacy with respect to the actual sensor data cannot be addressed by an abstract system, as it may highly depend on the specific sensing task. Moreover, some responsibility remains with the user, eg. to make sure transmitted photos do not contain something the user does not want to share.

However, the system can prevent profiling to some extend, eg. by providing report unlinkability (as PEPSICo does), ensuring records cannot be linked preventing tracking and identification of users based on their activity over time. Within this work, we focus on this aspect of privacy. However, we assume that linking transactions is not possible based on the network data alone.

Therefore, the network should not give away identifying data, eg. it should remove IP address and additional information that could be used to identify individual devices or to derive their location. If the network does not provide this property, anonymity services such as the Tor project [Tor19] could be used.

Participant attraction

The strength of participatory systems is that they allow for much larger and more diverse participation than traditional systems, where the data is gathered by a small research team. However, this requires that people are willing to participate. Some participatory systems assume altruistic motivation (cf. [HCG15]) and that might work for some research problems, especially if there is a large public awareness and the system minimizes the privacy risks of the participants. However, more participants might be attracted by adding an extrinsic motivation factor, which is usually done via an incentive mechanism. Such a mechanism rewards users for their participation, eg. by offering monetary compensation or other benefits. In general, participatory systems should, therefore, support such a mechanism.

Data trustworthiness

Because of the openness of participatory sensing systems, they are usually exposed to erroneous or even malicious data. For example, participants might unintentionally gather sensing data in the

wrong environment (eg. indoors instead of outdoors). Moreover, participants might maliciously report bad data. How to detect bad data is a difficult problem that highly depends on the sensing task. Including an incentive mechanism might even worsen the data trustworthiness as the submission of forged data reports can now result in a personal benefit for participants.

A general approach to increase the trustworthiness of the gathered data are reputation systems. Users get assigned reputation scores based on the quality of their contributions, as far as it can be determined. Those scores are then used as an indicator for the trustworthiness of future reports.

2.5 Related work

A lot of research has already been done in the area of privacy-preserving participatory sensing. However, most of this work requires strong trust assumptions, such as requiring multiple trusted parties or only considering honest-but-curious system operators, and does not provide formal security evaluations. An exception is PEPSICo [GMP14], which we extend in this work. The PEPSICo model is given in Chapter 4.

2.5.1 Surveys

There are not many surveys that provide an overview of all the approaches to address the privacy issue in participatory sensing systems. Christin [Chr15] surveyed privacy-preserving mechanisms based on a high-level threat model. They provide an extensive list of the mechanisms available in 2015 and classify them according to their threat models and used protection techniques. Moreover, they point out that pseudonymity is insufficient to protect the privacy of the participants and point out several studies that show how the identity of users could still be derived in such a case (eg. [Kru07; De +13]). There is also a predecessor survey from 2011 [Chr+11a].

In a more recent survey, Wang et al. [Wan+19] define a more detailed threat model for participatory sensing applications and evaluate numerous participatory sensing systems based on the privacy requirements they fulfill and the privacy goals they address.

2.5.2 Partial approaches

To protect the user's location information, spatial cloaking techniques are used by several systems. This mainly includes k-anonymity techniques, which requires that the user's location is indistinguishable from the location of at least $k - 1$ other users (cf. [SS98]). An example is PiRi [KS11], a peer-to-peer system where participants distribute tasks among each other to achieve k-anonymity during location-based sensing task assignment. TAPAS [KS12] further extends this solution to improve the quality of the submitted data. Nevertheless, in these solutions, privacy depends on the trustworthiness of other users. A centralized approach is given by Vu, Zheng, and Gao [VZG12] and Gao et al. [Gao+12], where a trusted party is used to cloak location information among the information of other users.

A different approach for location-based task assignment is given by [TGS14], where the network provider, which already knows the users' location, acts as a broker between them and the application

server. They use geocasting (cf. [NI97]) to deliver task requests to the users.

LOCATE [BK13] proposes a distributed model in the collective design and investigation setting, that is, users and queriers are identical. They focus on the privacy of location trajectories, which they preserve in opposite to many other systems, and narrow their approach to the Android operating system. Each user has a local database storing his sensed data but can also issue queries on data stored across the system. Their approach is to distribute the data trajectories among multiple user databases so that adversaries cannot decide which user a trajectory belongs to.

More general approaches to protect user information are based on data perturbation or aggregation. For example, Zhang et al. [Zha+12] propose the PESP algorithm, which adds dynamically varying noise to the user's data. Thus, they hide the actual values and even trends of individual data streams while still allowing global statistics to be computed. Moreover, Li and Cao [LC12; LC13] uses additively homomorphic encryption to construct a very efficient scheme where an aggregator colluding with a fraction of the users can only obtain the aggregate of all users data but not the individual values or intermediate results. They give protocols to compute the sum and the minimum of the submitted values, from which further protocols such as computing the average or maximum can be derived. However, because of their nature, such schemes require to trust the users to submit honest data.

To guarantee user anonymity, Christin et al. [Chr+11b] proposes to exchange sensor readings between participants that physically meet. In [Chr+14], they further improve their approach by assigning trust levels to participants, wherefore a minimum level requirement for data exchanges can be specified by the users.

Jin et al. [Jin+16] focuses on the privacy and economics of the incentive mechanism. They feature an auction-based model where the user's bid on a bundle of tasks by specifying their price for providing the data. Based on these bids, the platform selects a set of winners, and, after they send their data reports, they are given their payment. The scheme provides differential privacy for the bidding profiles of users.

"Worth One Minute" (WOM) [Klo+19] tries to decouple the incentive mechanism from the participatory sensing system by proposing an open and anonymous rewarding platform. However, they only consider anonymity against honest-but-curious system operators and do not formally evaluate the security of their system.

In opposite to the approaches listed above, PEPPeR [DKS12] does not protect the privacy of the users but the privacy of queriers. Queriers anonymously purchase tokens from the service provider that allow them to directly contact users and forward their queries to them. Moreover, the scheme provides double-spending detection, which requires users to contact a witness server before accepting a token.

2.5.3 General purpose frameworks

An early approach of a participatory sensing system protecting the participant's privacy is AnonySense [Shi+11]. It considers collusion attacks by other participants and queriers and some of the system components. Moreover, it considers the integrity of messages against malicious network access points. It uses Tor [Tor19] as an anonymizing network, together with the MIX network Mixmaster [Cot+08]

and an anonymization service that mixes sensitive reports with the reports from other mobile nodes and can additionally apply blurring techniques to further mask sensitive information. Moreover, group signatures are used to authenticate messages from registered mobile nodes. However, these mobile nodes are trusted not to submit invalid reports and only an informal security evaluation based on predefined attack scenarios is given.

IncogniSense [Chr+13] makes use of pseudonyms, which, however, are changed periodically to protect the privacy of the participants. It mainly focuses on its reputation mechanism and they use blind signatures to circumvent the need for a trusted third party, as it is used by Huang, Kanhere, and Hu [HKH12]. They use cloaking techniques to prevent the linkage of pseudonyms based on the evolution of their reputation scores. However, these techniques may incur a loss of reputation for users. Moreover, the time intervals in which the pseudonyms are changed have to be selected carefully as within one interval, all transactions are linkable. They only consider user privacy against honest-but-curious system operators in their security model and analyze the privacy properties of their system based on a predefined set of attacks instead of providing formal proofs.

SPPEAR [GGP14; GGP16] also uses frequently changed pseudonyms, however, they are changed after a specific number of contributions instead of a fixed time interval. They provide an incentive mechanism and accountability for malicious users. For the security of their system, they consider honest-but-curious system entities and malicious users. However, a collusion of some of their system operators can result in the full de-anonymization of users.

Lord of the Sense [MK14] is a model for the collective design and investigation approach. Therefore, it does not include an incentive mechanism but it includes a reputation mechanism. Users can vote for published reports which will affect the reputation of the user that submitted the report. Therefore, users have to contact a central server to update their reputation scores. The system is based on group signatures and provides anonymity against other users and accountability, that is, anonymity can be abolished if at least k users from the same group cooperate. However, the system components are trusted not to deviate from the protocol and the central server, where reports are submitted and reputation scores are updated, can link reports and, therefore, to profile users. Furthermore, only an informal security evaluation is given.

Privacy-Aware Incentivization (PAI) [Con+19] is a decentralized privacy-preserving participatory sensing system, featuring anonymous and unlinkable data reporting and incentive rewarding as well as an adaptive mechanism to compute incentive rewards. It adapts Identity Privacy Preserving Incentivization (IPPI) [CDB18], a decentralized and unlinkable incentive mechanism to achieve these goals. One of its advantages is that there is no direct communication between participants and queriers (which they call service providers in their model) and no central system operators, which prevent the tracing of participants via IP addresses. The threat model considers honest-but-curious queriers that use interference attacks to gain private information of participants. For submitted reports, the reported data is encrypted with the queriers public key and a one-time public encryption key of the user is attached. This ensures that only the querier can access the reported data. The incentive mechanism is controlled by the so-called OrderBook, which receives signed validity tokens from the querier, containing a unique

incentive identifier (ID) and the one-time public key of a user. It then publishes reward tokens which will be encrypted with the one-time key so that only the user that submitted the corresponding report can decrypt and spend those tokens. The anonymity of the users is ensured by this use of one-time keys in the incentive process. However, only informal security proofs are provided for the system.

3 Cryptographic preliminaries

In this chapter, we introduce the notation, security assumptions and cryptographic building blocks we use in this work.

3.1 Notation

Parties are denoted by calligraphic fonts (eg. $\mathcal{A}$, $\mathcal{U}$) and sets are denoted in fracture font (eg. $\mathfrak{C}$, $\mathfrak{I}$). Special sets are denoted in black board font (eg. $\mathbb{N}$, $\mathbb{Z}$). Algorithms and Protocols are denoted in teletype font (eg. `Setup`, `ReportData`). Oracles are usually denoted in standard font (eg. HonUser, CorruptRA).

To denote interactive protocols, the participating parties are noted in angular brackets with their corresponding inputs (eg. `Protocol` $\langle \mathcal{P}(input_{\mathcal{P}}), \mathcal{V}(input_{\mathcal{V}}) \rangle$ denotes the interactive protocol `Protocol` between $\mathcal{P}$ with input $input_{\mathcal{P}}$ and $\mathcal{V}$ with input $input_{\mathcal{V}}$). The parties are always denoted within the same order. If one of them is replaced, eg. by an adversary, the adversary acts in place of the party that is at the same position in the algorithm specification (eg. in `Protocol` $\langle \mathcal{A}(input_{\mathcal{A}}), \mathcal{V}(input_{\mathcal{V}}) \rangle$ $\mathcal{A}$ takes the role of $\mathcal{P}$ in the original protocol).

$\mathcal{A}^{\text{HonUser}}$ means that the party $\mathcal{A}$ is given access to the HonUser oracle. The notation $\mathcal{A}^{\mathsf{Dec}}(\mathsf{sk}, \cdot)$ specifies that $\mathcal{A}$ is given access to an oracle that executes the `Dec` algorithm using sk, which is unknown to $\mathcal{A}$, as its first input parameter and a value supplied by $\mathcal{A}$ as its second input parameter.

General

n, 1^n	Security parameter (standard and unary notation)
$\mathsf{negl}(n)$	An arbitrary function that is negligible in n
$\mathbb{N}$	Set of natural numbers
$\mathbb{Z}$	Set of integers
G_1, G_2, G_T	Cyclic groups, usually of prime order p

Parties

$\mathcal{A}$	Adversary	$\mathcal{AC}$	Accumulator (in BBA+)
$\mathcal{U}$	User	$\mathcal{RA}$	Registration Authority
$\mathcal{P}$	Prover	$\mathcal{SP}$	Service provider
$\mathcal{V}$	Verifier	$\mathcal{C}$	Challenger
$\mathcal{I}$	Issuer or Incentive System Provider (ISP)	$\mathcal{Q}$	Querier

3.2 Pseudorandom functions

A Pseudo Random Function (PRF) is an efficient distribution of functions where a key is used to select a specific function. As long as the key is randomly chosen, the selected function must be indistinguishable from a randomly chosen function from the set of all functions of the same form (cf. [KL07]).

Definition 3.1 (Pseudorandom function) *Let* $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *be an efficient, length-preserving, keyed function. We sad* F *is a* pseudorandom function *if for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{F_k(\cdot)}(1^n)\right] - \Pr\left[1 \leftarrow \mathcal{A}^{f_n(\cdot)}(1^n)\right]\right| \leq \mathsf{negl}(n)$$

where $k \leftarrow \{0,1\}^n$ *is chosen uniformly at random and* f_n *is chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings.*

3.3 Hash functions

The idea of a cryptographic hash function is to generate a unique, short fingerprint of arbitrarily large data. A hash function maps an input of arbitrary size to a bit string of a fixed size. By design, multiple possible inputs result in the same hash value. However, as these fingerprints should be unique, such *collisions* are required to be hard to find. This implies that hash functions are one-way functions (cf. [KL07]).

Definition 3.2 (Hash function) *A family of cryptographic hash functions is defined by a pair of algorithms* (HGen, H) *with the following properties:*

$\mathsf{hk} \xleftarrow{\$} \mathsf{HGen}(1^n)$
: *This probabilistic polynomial-time algorithm generates a hash key* hk *which identifies a specific function from the family.*

$h \leftarrow H_{\mathsf{hk}}(m)$
: *The keyed hash function is a function* $H_{\mathsf{hk}} : \{0,1\}^* \to \{0,1\}^{l(n)}$ *for a polynomial* l *that can be computed in deterministic polynomial-time.*

Definition 3.3 (Collision resistant hash function) *A family of hash functions* $\Pi = (\mathsf{HGen}, H)$ *is called* collision resistant *if for all* PPT *adversaries* $\mathcal{A}$ *and the experiment* $\mathrm{Exp}^{\mathrm{CR}}_{\Pi,\mathcal{A}}$ *defined in Figure 3.1, there exist a negligible function* negl *such that*

$$\Pr\left[\mathrm{Exp}^{\mathrm{CR}}_{\Pi,\mathcal{A}}(n) = 1\right] \leq \mathsf{negl}(n)$$

$\underline{\mathrm{Exp}^{\mathrm{CR}}_{\Pi,\mathcal{A}}(n)}$

$\mathsf{hk} \xleftarrow{\$} \mathsf{HGen}(1^n)$

$(m, m') \leftarrow \mathcal{A}(\mathsf{hk})$

return $m \neq m'$ and $H_{\mathsf{hk}}(m) \stackrel{?}{=} H_{\mathsf{hk}}(m')$

$\underline{\mathrm{Exp}^{\mathrm{PR}}_{\Pi,\mathcal{A}}(n)}$

$\mathsf{hk} \xleftarrow{\$} \mathsf{HGen}(1^n)$

$m \xleftarrow{\$} \{0,1\}^*; \quad h \leftarrow H_{\mathsf{hk}}(m)$

$(m') \leftarrow \mathcal{A}(\mathsf{hk}, h)$

return $H_{\mathsf{hk}}(m') \stackrel{?}{=} h$

Figure 3.1: Collision resistance and preimage resistance experiments for hash functions

Definition 3.4 (Preimage resistant hash function) *A family of hash functions* $\Pi = (\mathsf{HGen}, H)$ *is called* preimage resistant *if for all* PPT *adversaries* $\mathcal{A}$ *and the experiment* $\mathrm{Exp}^{\mathrm{PR}}_{\Pi,\mathcal{A}}$ *defined in Figure 3.1, there exists a negligible function* negl *such that*

$$\Pr\left[\mathrm{Exp}^{\mathrm{PR}}_{\Pi,\mathcal{A}}(n) = 1\right] \leq \mathsf{negl}(n)$$

Any family of hash functions that is collision resistant *is* preimage resistant*, that is*

$$\Pr\left[\mathrm{Exp}^{\mathrm{CR}}_{\Pi,\mathcal{A}}(n) = 1\right] \leq \mathsf{negl}(n) \implies \Pr\left[\mathrm{Exp}^{\mathrm{PR}}_{\Pi,\mathcal{A}}(n) = 1\right] \leq \mathsf{negl}(n)$$

3.4 Message authentication codes

As the name implies, Message Authentication Codes (MACs) are used to authenticate messages, that is to prevent an adversary from modifying a message sent by one party to another, or even injecting a new message in the communication, without the parties detecting the interference. They are the symmetric counterpart to digital signature schemes (Section 3.10). Thus, both parties are required to have a shared secret to authenticate legitimate messages. When sending a message, the sender uses this secret to compute a MAC tag on the message which can be validated using the same secret by the recipient. No adversary should be able to forge such a tag for any message not send by the legitimate parties before. We use the definition from [KL07].

Definition 3.5 (MACs) *A MAC is a tuple of* PPT *algorithms* $(\mathsf{Gen}, \mathsf{Mac}, \mathtt{Verify})$ *defined as follows:*

$\mathsf{k} \leftarrow \mathsf{Gen}(1^n)$

The key generation algorithm outputs a uniformly distributed key $k \in \{0,1\}^n$

$t \leftarrow \mathsf{Mac}(\mathsf{k}, m)$

The MAC tag generation algorithm generates a MAC tag for a message $m \in \{0,1\}^*$ *under the key* k

$\{0,1\} \leftarrow \mathtt{Verify}(\mathsf{k}, m, t)$

The MAC tag verification algorithm checks whether a tag t *is indeed a valid MAC tag for the message* m *under the key* k

A MAC is correct if honestly generated tags always verify, ie. if for all n, $\mathsf{k} \in \{0,1\}^n$ *and* $m \in \{0,1\}^*$ *it holds that*

$$\Pr\left[\mathtt{Verify}(\mathsf{k}, m, \mathtt{Mac}(\mathsf{k}, m)) \stackrel{?}{=} 1\right] = 1$$

$\mathrm{Exp}^{\text{MAC-forge}}_{\Pi,\mathcal{A}}(n)$

$\mathsf{k} \leftarrow \mathtt{kGen}(1^n)$

$(m, t) \leftarrow \mathcal{A}^{\mathtt{Mac}(\mathsf{k},\cdot)}(1^n)$

The experiment outputs 1 iff $\mathtt{Verify}(\mathsf{k}, m, t) \stackrel{?}{=} 1$ and $\mathcal{A}$ did not query $\mathtt{Mac}(\mathsf{k}, \cdot)$ for m during the experiment

Figure 3.2: Security experiment for MACs. Hereby, $\mathtt{Mac}(\mathsf{k}, \cdot)$ is an oracle that returns $\mathtt{Mac}(\mathsf{k}, m^*)$ for any m^* chosen by $\mathcal{A}$.

Definition 3.6 (Security of MACs) *Let* $\mathrm{Exp}^{\text{MAC-forge}}_{\Pi,\mathcal{A}}$ *be defined as in Figure 3.2. A message authentication code* $\mathrm{PI} = (\mathtt{Gen}, \mathtt{Mac}, \mathtt{Verify})$ *is called* existentially unforgable under adaptive chosen-message attacks (EUF-CMA secure) *if for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathrm{Exp}^{\text{MAC-forge}}_{\Pi,\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

3.5 Bilinear groups

Bilinear groups form the basis of pairing-based cryptography. We use the definition from [Har+19] ([Bel+09] for q-DDHI).

Definition 3.7 (Prime-order bilinear group generator) *A prime-order bilinear group generator is a* PPT *algorithm* SetupGrp *which behaves as follows:*

$gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathtt{SetupGrp}(1^n)$

Based on the security parameter n*, the group generator algorithm outputs the description of three cyclic groups* G_1*,* G_2*,* G_T *of prime order* p*, where* $\log p = \Theta(n)$*. Furthermore,* g_1 *and* g_2 *are the generators for* G_1 *and* G_2*, respectively.*

$e : G_1 \times G_2 \to G_T$

The pairing e *generated by* SetupGrp *efficiently maps a tuple* $(a, b) \in G_1 \times G_2$ *to an element of the target group* G_T *and has the following properties:*

Bilinearity

For all $a \in G_1$*,* $b \in G_2$*,* $x, y \in \mathbb{Z}_p$*, it holds that* $e(a^x, b^y) = e(a,b)^x y$*.*

Non-Degeneracy

$e(g_1, g_2)$ *generates* G_T*.*

Definition 3.8 (DDH and SXDH assumption) *The* DDH *assumption holds with respect to* SetupGrp *over G_i if for all* PPT *adversaries $\mathcal{A}$, there exist a negligible function* negl *such that*

$$\left|\Pr\left[b \stackrel{?}{=} b' \;\middle|\; \begin{array}{l} gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n) \\ x,y,z \leftarrow \mathbb{Z}_p \\ h_0 := g_i^{xy};\ h_1 := g_i^z \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(1^n, gp, g_i^x, g_i^y, h_b) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(n)$$

The SXDH *assumption holds with respect to* SetupGrp *is the* DDH *assumption holds for both source groups, ie. for both $i = 1$ and $i = 2$.*

Definition 3.9 (CDH assumption) *The* CDH *assumption holds with respect to* SetupGrp *over G_i if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[g_i^z \stackrel{?}{=} g_i^{xy} \;\middle|\; \begin{array}{l} gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n) \\ x,y \leftarrow \mathbb{Z}_p \\ z \leftarrow \mathcal{A}(1^n, gp, g_i^x, g_i^y) \end{array}\right] \leq \mathsf{negl}(n)$$

Obviously, the CDH *assumption is implied by the* DDH *assumption.*

Definition 3.10 (Co-CDH) *The* Co-CDH *assumption holds with respect to* SetupGrp *if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\Pr\left[a \stackrel{?}{=} g_2^x \;\middle|\; \begin{array}{l} gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n) \\ x \leftarrow \mathbb{Z}_p \\ a \leftarrow \mathcal{A}(1^n, gp, g_1^x) \end{array}\right] \leq \mathsf{negl}(n)$$

The Co-CDH *assumption is implied by the* SXDH *assumption.*

Definition 3.11 (q-DDHI) *The q-*DDHI *assumption holds with respect to* SetupGrp *over G_i if for all* PPT *adversaries $\mathcal{A}$, there exists a negligible function* negl *such that*

$$\left|\Pr\left[b \stackrel{?}{=} b' \;\middle|\; \begin{array}{l} gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n) \\ \alpha, r \leftarrow \mathbb{Z}_p^* \\ h_0 := g_i^{\frac{1}{\alpha}};\ h_1 := g_i^r \\ b \leftarrow \{0,1\} \\ b' \leftarrow \mathcal{A}(1^n, gp, g_i^{\alpha}, g_i^{\alpha^2}, g_i^{\alpha^3}, \ldots, g_i^{\alpha^q}, c_b) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(n)$$

3.6 Verifiable random functions

Verifiable Random Functions (VRFs) are a special kind of PRF (cf. Section 3.2) where, in addition, the party executing the function can compute a non-interactieve, publicly verifiable proof that the function was executed correctly (cf. [Bel+09]).

Definition 3.12 (VRF) *A VRF is a tuple of algorithms* $\mathrm{PI} = (\mathtt{SetupGrp}, \mathtt{SetupVRF}, \mathtt{Gen}, \mathtt{Prove}, \mathtt{Verify})$ *as follows*

$gp \leftarrow \mathtt{SetupGrp}(1^n)$
: *The group generation algorithm generates the public group parameters* gp *that define the message space of the scheme.*

$\mathsf{CRS} \leftarrow \mathtt{SetupVRF}(gp)$
: *This algorithm generates the Common Reference String (CRS)* CRS *of the scheme.*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{Gen}(\mathsf{CRS})$
: *The key generation algorithm generates the public key* pk *and the corresponding private key* sk*. For convenience, we assume that* CRS *is part of* pk*.*

$(y) \leftarrow \mathtt{Eval}(\mathsf{CRS}, \mathsf{sk}, x)$
: *This algorithm only evaluates the underlying PRF without computing a proof. It computes the pseudorandom image* y *of the preimage* x*.*

$(y, \pi) \leftarrow \mathtt{Prove}(\mathsf{CRS}, \mathsf{sk}, x)$
: *This algorithm computes the image* y *of* x *using the PRF and moreover, computes a proof* π *that allows to verify* y*.*

$\{0,1\} \leftarrow \mathtt{Verify}(\mathsf{pk}, x, y, \pi)$
: *This algorithm returns* 1 *if* π *is a valid proof for* y *being the image of* x *under the PRF and the private key* sk *corresponding to* pk*.*

Additionally, a VRF satisfies the following properties:

Correctness
: *Honestly computed proofs and images always verify, ie. for all* $n \in \mathbb{N}$ *and all* x *in the domain of the VRF it holds that*

$$\Pr\left[\, 1 \leftarrow \mathtt{Verify}(\mathsf{pk}, y, \pi) \;\middle|\; \begin{array}{l} gp \leftarrow \mathtt{SetupGrp}(1^n) \\ \mathsf{CRS} \leftarrow \mathtt{SetupVRF}(gp) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{Gen}(\mathsf{CRS}) \\ (y, \pi) \leftarrow \mathtt{Prove}(\mathsf{sk}, x) \end{array} \right]$$

$\underline{\mathrm{Exp}^{\text{VRF-ps}}_{\Pi,\mathcal{A}}(n)}$

$gp \leftarrow \mathtt{SetupGrp}(1^n)$
$\mathsf{CRS} \leftarrow \mathtt{SetupVRF}(gp)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{Gen}(\mathsf{CRS})$
$(x, state_0) \leftarrow \mathcal{A}_0^{\mathtt{Prove}(\mathsf{sk},\cdot)}(\mathsf{CRS}, \mathsf{pk})$
$(y_0, \pi_0) \leftarrow \mathtt{Prove}(\mathsf{sk}, x)$
$y_1 \leftarrow D(\mathsf{CRS})$
$b' \leftarrow \mathcal{A}_1^{\mathtt{Prove}(\mathsf{sk},\cdot)}(y_b, state_0)$

The experiment returns 1, iff $b \stackrel{?}{=} b'$ and $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ did not query $\mathtt{Prove}(\mathsf{sk}, \cdot)$ for x during the experiment and $D(\mathsf{CRS})$ is the domain of the VRF defined by CRS

Figure 3.3: Preudorandomness experiment for VRFs. Hereby, $\mathtt{Prove}(\mathsf{sk}, \cdot)$ is an oracle that computes the output of Prove under sk an input provided by the adversary

Pseudorandomness

The VRF is pseudorandom*, if for all* $n \in \mathbb{N}$ *and for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*, it holds that*

$$\left|\Pr\left[\mathrm{Exp}^{\text{VRF-ps}}_{\Pi,\mathcal{A}}(n) \stackrel{?}{=} 1\right]\right| \leq \mathsf{negl}(n)$$

Uniqueness

For all $n \in \mathbb{N}$, $gp \leftarrow \mathtt{SetupGrp}(1^n)$ *and* $\mathsf{CRS} \leftarrow \mathtt{SetupVRF}(gp)$*, there do not exist* $(\mathsf{pk}, x, y_0, y_1, \pi_0, \pi_1)$ *such that*

$$y_0 \neq y_1 \wedge \mathtt{Verify}(\mathsf{pk}, x, y_0, \pi_0) \stackrel{?}{=} \mathtt{Verify}(\mathsf{pk}, x, y_1, \pi_1) \stackrel{?}{=} 1$$

3.7 Commitments

A commitment scheme allows committing to a message which remains secret until the commitment is opened (ie. the commitment is *hiding*). Moreover, the commitment can only be opened to the message it has been created for (ie. the commitment is *binding*). We use the definition from Hartung et al. [Har+17].

Definition 3.13 (F_{gp}-binding Commitment) *A non-interactive commitment scheme is a tuple of* PPT *algorithms* $(\mathtt{SetupGrp}, \mathtt{Gen}, \mathtt{Com}, \mathtt{Open})$*. Let* $F_{gp} : \mathfrak{M} \to \mathfrak{I}$ *be a bijective function mapping the message space* $\mathfrak{M}$ *to the implicit message space* $\mathfrak{I}$*. In an* F_{gp}*-binding commitment scheme, one commits to a message* $m \in \mathfrak{M}$ *but opens a commitment to* $F_{gp}(m) \in \mathfrak{I}$*.*

$gp \leftarrow \mathtt{SetupGrp}(1^n)$

The group setup algorithm generates the public group parameters gp that define the message space of the scheme.

$\mathsf{CRS} \leftarrow \mathtt{Gen}(gp)$

This algorithm generates the public CRS CRS*.*

$(com, d) \leftarrow \mathsf{Com}(\mathrm{CRS}, m)$

The commitment algorithm takes the CRS CRS *and the message* m *as input and outputs the commitment* c *and a decommitment value* d*.*

$\{0,1\} \leftarrow \mathsf{Open}(\mathrm{CRS}, com, d, M)$

The opening algorithm returns 1 *if the commitment* com *can indeed be opened to* $M \in \mathfrak{I}$ *using the decommitment value* d *and* 0 *otherwise.*

The commitment scheme is called correct*, if* Open *always verifies honestly generated commitments, ie.*

$$\Pr\left[\mathsf{Open}(\mathrm{CRS}, com, d, F_{gp}(m)) \stackrel{?}{=} 1 \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathrm{CRS} \leftarrow \mathsf{Gen}(gp) \\ (com, d) \leftarrow \mathsf{Com}(\mathrm{CRS}, m) \end{array}\right] = 1$$

The commitment scheme is called hiding *if an adversary cannot distinguish between commitments to two message of his choice with more than negligible advantage, ie. for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$

$$\left|\Pr\left[b = b' \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathrm{CRS} \leftarrow \mathsf{Gen}(gp) \\ (m_0, m_1, state_0) \leftarrow \mathcal{A}_0(\mathrm{CRS}), b \stackrel{\$}{\leftarrow} \{0,1\} \\ com \leftarrow \mathsf{Com}(\mathrm{CRS}, m_b); b' \leftarrow \mathcal{A}_1(\mathrm{CRS}, com, state_0) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(n)$$

The commitment scheme is called F_{gp}-binding *an adversary has at most a negligible probability of finding a commitment that can be opened to two different implicit messages, ie.*

$$\Pr\left[\begin{array}{l} \mathsf{Open}(\mathrm{CRS}, com, d, M) \stackrel{?}{=} 1 \\ \wedge\, \mathsf{Open}(\mathrm{CRS}, com, d', M') \stackrel{?}{=} 1 \\ \wedge\, M \neq M' \end{array} \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathrm{CRS} \leftarrow \mathsf{Gen}(gp) \\ (M, M', d, d') \leftarrow \mathcal{A}(\mathrm{CRS}) \end{array}\right] \leq \mathsf{negl}(n)$$

The commitment scheme is called additively homomorphic *if given a commitment to* m_0 *and a commitment to* m_1*, a commitment to their sum* $m_0 + m_1$ *can be efficiently computed without knowing* m_0 *and* m_1*. More precisely, there exist two* PPT *algorithms* CAdd *and* DAdd*, such that*

$$\Pr\left[\mathsf{Open}(\mathrm{CRS}, com, d, m_0 + m_1) \stackrel{?}{=} 1 \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathrm{CRS} \leftarrow \mathsf{Gen}(gp) \\ (com_0, d_0) \leftarrow \mathsf{Com}(\mathrm{CRS}, m_0) \\ (com_1, d_1) \leftarrow \mathsf{Com}(\mathrm{CRS}, m_1) \\ com \leftarrow \mathsf{CAdd}(com_0, com_1); d \leftarrow \mathsf{DAdd}(d_0, d_1) \end{array}\right] \stackrel{?}{=} 1$$

The commitment scheme is called equivocable *if there is a trapdoor for the CRS that allows to open it*

to any given implicit message, ie. there exist PPT *algorithms* SimGen, SimCom *and* Equiv *such that for all* PPT *adversaries the following holds:*

1. *A CRS generated by* Gen *is statistically indistinguishable from a CRS generated by* SimGen*, together with a simulation trapdoor* $\mathsf{td}_{\mathsf{com}}$*, ie.*

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}) \;\middle|\; \begin{array}{l} gp \leftarrow \mathtt{SetupGrp}(1^n) \\ \mathrm{CRS} \leftarrow \mathtt{Gen}(gp) \end{array}\right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}') \;\middle|\; \begin{array}{l} gp \leftarrow \mathtt{SetupGrp}(1^n) \\ (\mathrm{CRS}', \mathsf{td}_{\mathsf{com}}) \leftarrow \mathtt{SimGen}(gp) \end{array}\right] \end{array} \right| \stackrel{?}{=} 0$$

2. *With trapdoor* $\mathsf{td}_{\mathsf{com}}$ *a commitment can be generated that can latter be opened to any message within the message space, ie.*

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}', \mathsf{td}_{\mathsf{com}}, m, com, d) \;\middle|\; \begin{array}{l} gp \leftarrow \mathtt{SetupGrp}(1^n) \\ (\mathrm{CRS}', \mathsf{td}_{\mathsf{com}}) \leftarrow \mathtt{SimGen}(gp) \\ m \leftarrow \mathfrak{M} \\ (com, d) \leftarrow \mathtt{Com}(\mathrm{CRS}', m) \end{array}\right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}', \mathsf{td}_{\mathsf{com}}, m, com', d') \;\middle|\; \begin{array}{l} gp \leftarrow \mathtt{SetupGrp}(1^n) \\ (\mathrm{CRS}', \mathsf{td}_{\mathsf{com}}) \leftarrow \mathtt{SimGen}(gp) \\ (com', r) \leftarrow \mathtt{SimCom}(gp) \\ m \leftarrow \mathfrak{M} \\ d' \leftarrow \mathtt{Equiv}(\mathrm{CRS}', \mathsf{td}_{\mathsf{com}}, m, r) \end{array}\right] \end{array} \right| \leq \mathsf{negl}(n)$$

3.8 Public key encryption

Encryption is used to hide the information contained within a message from everyone but the intended recipients. In opposite to symmetric or private key encryption, in an asymmetric or Public Key Encryption (PKE) scheme, there are separate keys for encryption and decryption. A message is first encrypted with the public key of the recipient, which as the name suggests, can publicly available and does not allow the decryption of messages. However, messages can be decrypted using the corresponding private key that should only be available to the recipient. We use the definition from BBA+ [Har+17].

Definition 3.14 (PKE) *A PKE scheme is a tuple of* PPT *algorithms* (SetupGrp, Gen, Enc, Dec) *defined as follows:*

$gp \leftarrow \mathtt{SetupGrp}(1^n)$

The group setup algorithm generates the public group parameters gp*.*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(gp)$

This key generation algorithm outputs the public encryption key pk *and the corresponding private decryption key* sk.

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$

The encryption algorithm takes as input a public key pk *and a message m and outputs the ciphertext c.*

m or $\perp \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$

The deterministic decryption algorithm takes as input a private key sk *and a ciphertext c and outputs the message m hidden in c if c is a valid encryption of m under the public key* pk *corresponding to* sk*. Otherwise, the algorithm outputs* $\perp$*, indicating failure.*

A PKE scheme is correct *if for all* $n \in \mathbb{N}$, $gp \leftarrow \mathsf{SetupGrp}(1^n)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(gp)$ *and message m from the message space of the scheme, it holds that*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \stackrel{?}{=} m$$

$\underline{\mathrm{Exp}^{\text{IND-CCA}}_{\Pi,\mathcal{A}}(n)}$

$gp \leftarrow \mathsf{SetupGrp}(1^n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(gp)$
$(m_0, m_1, state_0) \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$, where $|m_0| \stackrel{?}{=} |m_1|$
$b \leftarrow \{0,1\}$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b, state_0)$
$b' \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(c)$

The experiment returns 1 iff $b \stackrel{?}{=} b'$ and $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ did not query $\mathsf{Dec}(\mathsf{sk}, \cdot)$ for c

Figure 3.4: IND-CCA experiment for PKE schemes. $\mathsf{Dec}(\mathsf{sk}, \cdot)$, is an oracle that returns $\mathsf{Dec}(\mathsf{sk}, c^*)$ for an c^* chosen by $\mathcal{A}$

Definition 3.15 (Security of PKE) *Let* Π *be a PKE scheme and* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *a* PPT *adversary. We define the* IND-CCA *experiment in Figure 3.4.* Π *is called* IND-CCA *secure and therefore provides* indistinguishability under chosen-ciphertext attacks*, if for all* PPT $\mathcal{A}$

$$\Pr\left[\mathrm{Exp}^{\text{IND-CCA}}_{\Pi,\mathcal{A}}(n)\right] \leq \mathsf{negl}(n)$$

A weaker notion of for the security of a PKE scheme is indistinguishability under chosen-plaintext attacks *(*IND-CPA*). It is defined equivalent to* IND-CCA*, but in the* IND-CPA *experiment* $\mathcal{A}$ *has no access to the* Dec *oracle.*

3.9 Identity-based encryption

The idea of Identity-based Encryption (IBE) was first introduced by Shamir [Sha84]. An IBE scheme is similar to a PKE scheme, except that an arbitrary string, such as a user's identity, can be used as the public key. However, if every member of the system could be able to compute the secret key for a given string, the scheme would not provide any security. Therefore, a Trusted Third Party (TTP) is required in the scheme. The TTP has additional information (the so-called master secret key) which allows the efficient computation of user secret keys. The TTP is responsible for authenticating the users before they are given their user secret key.

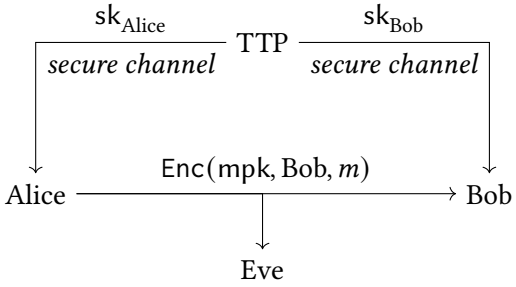

Figure 3.5: Sending an encrypted message in an IBE scheme

Figure 3.5 shows how such a system would work with two users, Alice and Bob. First, the secret keys are distributed by the TTP via a secure channel. Using Bob's identity, Alice can now send him encrypted messages that hide all information about the message (except its length) from potential eavesdroppers. We use the same definition as in PEPSICo [GMP14], which is based upon [Abd+05] and [BF01].

Definition 3.16 (IBE) *An IBE scheme is a tuple of algorithms* (Setup, Extract, Enc, Dec)*:*

$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathtt{Setup}(1^n)$

The setup algorithm generates the master key pair. The master public key mpk *together with the identity of the recipient is required for the encryption. The master secret key* msk *allows to extract the secret decryption key for a specific identity.*

$\mathsf{sk}_{id} \xleftarrow{\$} \mathtt{Extract}(\mathsf{mpk}, \mathsf{msk}, id)$

This algorithm extracts the secret decryption key sk_{id} *for the identity* $id \in \{0,1\}^*$.

$c \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, id, m)$

The encryption algorithm encrypts a message m *from the message space* $\mathfrak{M}$ *under an identity* id*, resulting in a ciphertext* c *in the cipher-space* $\mathfrak{C}$*.*

$m \leftarrow \mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{id}, c)$

The decryption algorithm decrypts the ciphertext $c \in \mathfrak{C}$ *with the help of the corresponding secret key* sk_{id}*, resulting in a message* $m \in \mathfrak{M}$*.*

An IBE scheme is called correct*, if for all* $id \in \{0,1\}^*$ *and for all* $m \in \mathfrak{M}$

$$\mathsf{Dec}(\mathsf{mpk}, \mathtt{Extract}(\mathsf{mpk}, \mathsf{msk}, id), \mathsf{Enc}(\mathsf{mpk}, id, m)) = m$$

Definition 3.17 (Security of IBE) *Let* Π *be an IBE scheme and* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *a* PPT *adversary. The* ANO-IND-ID-CCA *experiment is defined in Figure 3.6.*

$\mathrm{Exp}^{\text{ANO-IND-ID-CCA}}_{\Pi,\mathcal{A}}(n)$

$(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathtt{Setup}(1^n)$

$((id_0, m_0), (id_1, m_1), state_0) \leftarrow \mathcal{A}_0^{\mathtt{Extract}(\mathsf{mpk},\mathsf{msk},\cdot),\mathsf{Dec}(\mathsf{sk}_{id},\cdot)}(\mathsf{mpk})$

$b \xleftarrow{\$} \{0,1\}$

$c \leftarrow \mathsf{Enc}(\mathsf{mpk}, id_b, m_b)$

$b' \leftarrow \mathcal{A}_1^{\mathtt{Extract}(\mathsf{mpk},\mathsf{msk},\cdot),\mathsf{Dec}(\mathsf{sk}_{id},\cdot)}(\mathsf{mpk}, c, state_0)$

The experiment returns 1 iff $b \stackrel{?}{=} b'$ and $\mathcal{A}$ did not query $\mathtt{Extract}(\mathsf{mpk}, \mathsf{msk}, \cdot)$ on id_0 or id_1 nor $\mathsf{Dec}(\mathsf{sk}_{id}, \cdot)$ on sk_{id_0} or sk_{id_1} and the challenge c.

Figure 3.6: ANO-IND-ID-CCA experiment for IBE. Hereby, $\mathtt{Extract}(\mathsf{mpk}, \mathsf{msk}, \cdot)$ is an oracle that returns $\mathtt{Extract}(\mathsf{mpk}, \mathsf{msk}, id^*)$ for any id^* chosen by $\mathcal{A}$. $\mathsf{Dec}(\mathsf{sk}_{id}, \cdot)$ is an oracle that returns $\mathsf{Dec}(\mathsf{sk}_{id^*}, c^*)$ for any id^*, c^* chosen by the adversary.

The IBE scheme Π *is called* ANO-IND-ID-CCA *secure and therefore provides* anonymity and indistinguishability under chosen-ciphertext attacks*, if all adversaries can win the experiment with at most negligible advantage, ie. for all* PPT *adversaries* $\mathcal{A}$

$$\mathrm{Adv}^{\text{ANO-IND-ID-CCA}}_{\Pi,\mathcal{A}}(n) := \left| \Pr\left[\mathrm{Exp}^{\text{ANO-IND-ID-CCA}}_{\Pi,\mathcal{A}}(n) = 1\right] - \frac{1}{2} \right| \leq \mathrm{negl}(n)\,.$$

For $id_0 = id_1$*, the experiment models only* indistinguishability *(*IND-ID-CCA *security) and for* $m_0 = m_1$ *only* anonymity *(*ANO-ID-CCA *security). If* $\mathcal{A}$ *is not granted access to the* Dec *oracle, the experiment models the respective* chosen-plaintext *security variants instead (*ANO-IND-ID-CPA, IND-ID-CPA *and* ANO-ID-CPA *security).*

3.10 Digital signatures

Digital signatures were designed as an analogy to handwritten signatures. They are used to authenticate messages and validate their integrity. Similar to asymmetric encryption, there is a public and private key pair. The private signing key can be used to generate a signature for a message. Given the message and the signature, the public verification key can be used to validate that the signature is valid for the message and therefore has been generated by the owner of the private key.

Definition 3.18 (Signatures) *A signature scheme is a tuple of* PPT *algorithms* (SetupGrp,Gen,Sign, Verify) *satisfying the following*

$gp \leftarrow \mathtt{SetupGrp}(1^n)$
: *The group setup algorithm generates the public group parameters* gp*.*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{Gen}(gp)$
: *This key generation algorithm outputs the public verification key* pk *and the corresponding private signing key* sk*.*

$\sigma \leftarrow \mathtt{Sign}(\mathsf{sk}, m)$
: *The signing algorithm takes as input a private key* sk *and a message* m *and outputs the corresponding signature* σ*.*

$\{0,1\} \leftarrow \mathtt{Verify}(\mathsf{pk}, m, \sigma)$
: *The deterministic verification algorithm takes as input a public key* pk*, am message* m *and a signature* σ*. It outputs* 1 *if* σ *is a valid signature for* m *for the given* pk *and* 0 *otherwise.*

A signature scheme is correct *if for all* $n \in \mathbb{N}$, $(gp) \leftarrow \mathtt{SetupGrp}(1^n)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{Gen}(gp)$ *and message* m *from the message space of the scheme*

$$\mathtt{Verify}(\mathsf{pk}, m, \mathtt{Sign}(\mathsf{sk}, m)) \stackrel{?}{=} 1$$

Definition 3.19 (Security of signature schemes) *Let* Π *be an signature scheme and* $\mathcal{A}$ *be an* PPT *adversary. Then, the* EUF-CMA *experiment is defined in Figure 3.7.*

$\mathrm{Exp}^{\text{EUF-CMA}}_{\Pi,\mathcal{A}}(n)$

$gp \leftarrow \mathtt{SetupGrp}(1^n)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{Gen}(gp)$
$(m, \sigma) \leftarrow \mathcal{A}^{\mathtt{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$

The experiment returns 1 iff $\mathtt{Verify}(\mathsf{pk}, \sigma) \stackrel{?}{=} 1$ and $\mathcal{A}$ did not query $\mathtt{Sign}(\mathsf{sk}, \cdot)$ for m.

Figure 3.7: EUF-CMA experiment for digital signature schemes. Hereby, $\mathtt{Sign}(\mathsf{sk}, \cdot)$ is an oracle that returns $\mathtt{Sign}(\mathsf{sk}, m^*)$ for any m^* chosen by the adversary.

Π *is called* existentially unforgeable under adaptive chosen-message attacks (EUF-CMA) *secure, if for all* PPT *adversaries* $\mathcal{A}$

$$\Pr\left[\mathrm{Exp}^{\text{EUF-CMA}}_{\Pi,\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

3.11 Non-interactive zero-knowledge proof systems

A zero-knowledge proof system allows a prover to convince a verifier of the truth of a statement, without the verifier learning anything beyond that fact. For a non-interactive proof system, the proof consists of a single message sent from the prover to the verifier, without further interactions between those parties. We use the definition of F_{gp}-extractable Non-interactive Zero-Knowledge proof systems (NIZKs) from [Har+19].

Definition 3.20 (F_{gp}-extractable NIZKs) *Let R be an efficiently verifiable relation containing triples (gp, stm, wit). We call gp the group setup, stm the statement and wit the witness. Let L_{gp} be the language containing all statements stm such that $(gp, stm, wit) \in R$. Then a non-interactive zero-knowledge proof system for R is a tuple of* PPT *algorithms* $\mathsf{pok} := (\mathtt{SetupGrp}, \mathtt{SetupPoK}, \mathtt{Prove}, \mathtt{Verify})$.

$gp \leftarrow \mathtt{SetupGrp}(1^n)$
: *The group setup algorithm generates the public group parameters gp.*

$\mathsf{CRS} \leftarrow \mathtt{SetupPoK}(gp)$
: *This algorithm generates the CRS for the proof system. We assume the CRS contains the public group parameters gp.*

$\pi \leftarrow \mathtt{Prove}(\mathsf{CRS}, stm, wit)$
: *The proof generation algorithm outputs a proof π if wit is a valid witness for the fact that stm is within the language L_{gp}, ie. $(gp, x, w) \in R$*

$\{0,1\} \leftarrow \mathtt{Verify}(\mathsf{CRS}, stm, \pi)$
: *The verify algorithm checks if π is a valid proof attesting that the statement stm is contained within L_{gp}.*

The proof system is perfectly complete *if all honestly generated proofs verify, ie. for all $n \in \mathbb{N}$, $gp \leftarrow \mathtt{SetupGrp}(1^n)$, $\mathsf{CRS} \leftarrow \mathtt{SetupPoK}(gp)$, $(gp, stm, wit) \in R$ and $\pi \leftarrow \mathtt{Prove}(\mathsf{CRS}, stm, wit)$ we have that*

$$\mathtt{Verify}(\mathsf{CRS}, stm, \pi) \stackrel{?}{=} 1$$

The proof system is perfectly sound *if no (possibly unbounded) adversary can forge a proof for a false statement, ie. for all adversaries $\mathcal{A}$*

$$\Pr\left[\mathtt{Verify}(\mathsf{CRS}, stm, \pi) \stackrel{?}{=} 1 \;\middle|\; \begin{array}{l} gp \leftarrow \mathtt{SetupGrp}(1^n) \\ \mathsf{CRS} \leftarrow \mathtt{SetupPoK}(gp) \\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{CRS}) \\ x \notin L_{gp} \end{array}\right] \stackrel{?}{=} 0$$

The proof system is perfectly F_{gp}-extractable *if we can replace the CRS with an extraction CRS which*

allows to extract $F_{gp}(wit)$ *using a trapdoor* $\mathsf{td}_{\mathsf{epok}}$. *More precisely, there exists a* PPT *extractor* (SetupEPoK, ExtractW) *such that for all (possibly unbounded) adversaries* $\mathcal{A}$ *it holds that*

1. $\mathcal{A}$ *cannot distinguish between a* CRS *generated by* SetupPoK *and one generated by* SetupEPoK, *ie.*

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}) \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathrm{CRS} \leftarrow \mathsf{SetupPoK}(gp) \end{array}\right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}') \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ (\mathrm{CRS}', \mathsf{td}_{\mathsf{epok}}) \leftarrow \mathsf{SetupEPoK}(gp) \end{array}\right] \end{array} \right| \stackrel{?}{=} 0$$

2. *From each valid proof,* $F_{gp}(wit)$ *can be extracted for a valid witness* wit, *ie.*

$$\Pr\left[\begin{array}{l} \exists wit : F_{gp}(w) \stackrel{?}{=} W \\ \qquad \wedge (gp, stm, wit) \in R \end{array} \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ (\mathrm{CRS}', \mathsf{td}_{\mathsf{epok}}) \leftarrow \mathsf{SetupEPoK}(gp) \\ (stm, \pi) \leftarrow \mathcal{A}(\mathrm{CRS}') \\ 1 \leftarrow \mathsf{Verify}(\mathrm{CRS}', stm, \pi) \\ W \leftarrow \mathsf{ExtractW}(\mathrm{CRS}', \mathsf{td}_{\mathsf{epok}}, stm, \pi) \end{array}\right] \stackrel{?}{=} 1$$

The proof system is composable zero-knowledge *if the CRS can be replaced with a simulation CRS that allows to simulate proofs without the knowledge of a witness. More precisely, there exists a* PPT *simulator* (SetupSPoK, SimProof) *and hint generator* GenHint *such that for all* PPT *adversaries it holds that*

1. $\mathcal{A}$ *cannot distinguish between a* CRS *generated by* SetupPoK *and one generated by* SetupSPoK, *ie.*

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}) \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ \mathrm{CRS} \leftarrow \mathsf{SetupPoK}(gp) \end{array}\right] \\ -\Pr\left[1 \leftarrow \mathcal{A}(\mathrm{CRS}') \;\middle|\; \begin{array}{l} gp \leftarrow \mathsf{SetupGrp}(1^n) \\ hint \leftarrow \mathsf{GenHint}(gp) \\ (\mathrm{CRS}', \mathsf{td}_{\mathsf{spok}}) \leftarrow \mathsf{SetupSPoK}(gp, hint) \end{array}\right] \end{array} \right| \leq \mathsf{negl}(n)$$

2. $\mathcal{A}$ *cannot distinguish between real and simulated proofs with more than negligible probability, ie. for all* $gp \leftarrow \mathsf{SetupGrp}(1^n)$ *and* $(\mathrm{CRS}', \mathrm{td}) \leftarrow \mathsf{SetupSPoK}(gp)$

$$\left| \begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Prove}(\mathrm{CRS}', \cdot, \cdot)}(1^n, \mathrm{CRS}', \mathsf{td}_{\mathsf{spok}})\right] \\ -\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{SimProof}'(\mathrm{CRS}', \mathsf{td}_{\mathsf{spok}}, \cdot, \cdot)}(1^n, \mathrm{CRS}', \mathsf{td}_{\mathsf{spok}})\right] \end{array} \right| \leq \mathsf{negl}(n)$$

where SimProof′ *is an oracle which on input* $(stm, wit) \in R$ *returns* $\mathsf{SimProof}(\mathrm{CRS}', \mathsf{td}_{\mathsf{spok}}, stm)$. *On input* $(stm, wit) \notin R$ SimProof′ *and* Prove *both return* $\perp$.

4 PEPSICo

The Privacy-enhanced Participatory Sensing Infrastructure (PEPSI) was introduced by De Cristofaro and Soriente [DS11; DS13] as an model for privacy-preserving participatory sensing. Günther, Manulis, and Peter [GMP14] addressed some of its shortcomings (mainly collusion resistance) and published a revised version as the PEPSICo model.

In this chapter, we describe the PEPSICo model and the security properties it defines as it is fundamental to this work.

4.1 The PEPSICo model

The PEPSICo model consists of the following parties [GMP14]:

Mobile Nodes (MNs)
: These are the actual sensor devices, eg. smartphones that are owned by the participants

Querier
: These are the entities interested in receiving sensor reports

Service Provider (SP)
: The SP acts as a broker between the MNs and the queriers. It stores the data reports of the MNs and forwards them to the queriers.

Registration Authority (RA)
: The RA performs the system setup and handles the registration of mobile nodes and queriers.

The architecture of the model is shown in Figure 4.1. MNs and queriers have to register to the RA before they can use the system. After registration, a MN can send its data reports to the SP, where individual reports can optionally be aggregated. A querier can subscribe for reports with the SP.

Based on this infrastructure, the model formulates privacy and security notions to protect the Mobile Nodes and queriers. The basic idea is that the query identity describing the sensing task as well as the report data are hidden from the Service Providers and individual data reports are not linkable to a MN's identity.

Definition 4.1 (PEPSICo) *A PEPSICo instantiation consists of a tuple of algorithms (*`Setup`, `RegisterMN`, `RegisterQ`, `ReportData`, `SubscribeQuery`, `ExecuteQuery`, `DecodeData`*). Optionally, an additional* `AggregateData` *algorithm can be defined.*

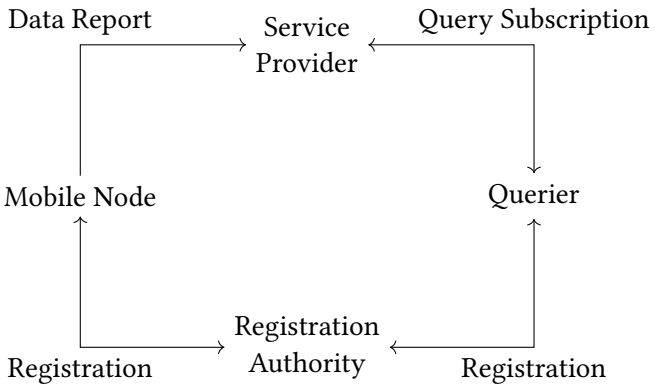

Figure 4.1: The PEPSICo infrastructure

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \xleftarrow{\$} \mathtt{Setup}(1^n)$

The setup of the scheme is executed by the RA and outputs its public key $\mathsf{pk}_{\mathcal{RA}}$ *and secret key* $\mathsf{sk}_{\mathcal{RA}}$. $\mathsf{pk}_{\mathcal{RA}}$ *contains a description of the query identity space* $\mathfrak{I}$ *and the message space* $\mathfrak{M}$.

$\mathsf{regMN}_{qid} \xleftarrow{\$} \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

The MN registration is executed by the RA when a new MN wants to contribute data for a given query identity $qid \in \mathfrak{I}$. *It outputs the mobile node registration value* regMN_{qid} *which the RA has to send to the MN.*

$\mathsf{regQ}_{qid} \xleftarrow{\$} \mathtt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

The querier registration algorithm is executed by the RA to register a new querier for a given query identity $qid \in \mathfrak{I}$. *The querier registration value* regQ_{qid} *has to be sent to the querier.*

$c \xleftarrow{\$} \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$

The data report algorithm is executed by the MN to report some data $m \in \mathfrak{M}$ *under a query identity* $qid \in \mathfrak{I}$. *The output of the algorithm is a data report* c *which the mobile node has to sent to the SP.*

$s \xleftarrow{\$} \mathtt{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid)$

The query subscription algorithm is executed by the querier to generate a subscription token s *for a given query identity* $qid \in \mathfrak{I}$. *The subscription token has to be send to the SP.*

c or $\perp \xleftarrow{\$} \mathtt{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, s)$

The query execution algorithm is executed by the SP to determine whether the hidden query identity of a given report c *matches with the subscription token* s. *In case of a match, the algorithm outputs* c *to indicate that the report has to be sent to the querier subscribing for* s, *else* $\perp$.

m or $\perp \xleftarrow{\$} \mathtt{DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid, c)$

The decoding algorithm is executed by the querier to obtain the data m *from a received data report* c. *In case that the query identity does not match with the report, the algorithm outputs* $\perp$.

c or $\perp \xleftarrow{\$} \texttt{AggregateData}(\mathsf{pk}_{\mathcal{RA}}, \vec{c})$

The optional data aggregation algorithm is executed by the SP to combine multiple data reports $\vec{c} = (c_0, \ldots, c_k)$. If all of them have been created for the same query identity, the output is a single, aggregated data report c. Else the algorithm outputs $\perp$ to indicate failure.

A PEPSICo instantiation is called sound *if data reports match with query subscriptions and are decodable using the querier registration value generated for the same query identity. More precisely, for all $n \in \mathbb{N}$, $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \texttt{Setup}(1^n)$, $qid \in \mathfrak{I}$ and $m \in \mathfrak{M}$*

$$\Pr\left[\begin{array}{l|l} & \mathsf{regMN}_{qid} \leftarrow \texttt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid) \\ c \leftarrow \texttt{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, s) & c \leftarrow \texttt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m) \\ m \leftarrow \texttt{DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid, c) & \mathsf{regQ}_{qid} \leftarrow \texttt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid) \\ & s \leftarrow \texttt{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid) \end{array}\right] \stackrel{?}{=} 1$$

For a PEPSICo instantiation that provides the `AggregateData` *algorithm, it has to hold that for all* $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \texttt{Setup}(1^n)$, $qid \in \mathfrak{I}$ *and* $\vec{m} = (m_0, ..., m_k) \in \mathfrak{M}^k$

$$\Pr\left[\begin{array}{l|l} & \mathsf{regMN}_{qid} \leftarrow \texttt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid) \\ & \forall i \in 0, ..., k : c_i \leftarrow \texttt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m_i) \\ c \leftarrow \texttt{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, s) & c \leftarrow \texttt{AggregateData}(\mathsf{pk}_{\mathcal{RA}}, (c_0, ..., c_k)) \\ m' \stackrel{?}{=} \sum_0^k (m_k) & \mathsf{regQ}_{qid} \leftarrow \texttt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid) \\ & s \leftarrow \texttt{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid) \\ & m' \leftarrow \texttt{DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid, c) \end{array}\right] \stackrel{?}{=} 1$$

4.2 Security of PEPSICo

The adversary model of PEPSICo assumes confidential channels and allows the adversary to be a collusion of the parties, in particular of the SP and some MNs and queriers against other MNs and queriers.

To model the security of a PEPSICo instantiation PI, a PPT adversary $\mathcal{A}$ is considered that can corrupt other parties, in special cases even the RA. Therefore, the security games give the adversary access to a subset of the oracles described in Figure 4.2. $\mathfrak{CI}_{\text{MN}}$ and $\mathfrak{CI}_{Q}$ denote the set of identities that have been corrupted by $\mathcal{A}$. $\mathfrak{C}_{\mathcal{RA}}$ and $\mathfrak{C}_{\mathcal{SP}}$ denote whether the RA or the SP have been corrupted. They are initialized with 0 and set to 1 in the case of a corruption of the respective party.

There security and privacy of PEPSICo is defined by three properties, *Node privacy*, *query privacy* and *report unlinkablity*, which is defined in the following sections.

CorruptMN(qid)

$\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$\mathfrak{CI}_{\mathrm{MN}} \leftarrow \mathfrak{CI}_{\mathrm{MN}} \cup qid$
return regMN_{qid}

CorruptQ(qid)

$\mathsf{regQ}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$\mathfrak{CI}_{\mathcal{Q}} \leftarrow \mathtt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
return regQ_{qid}

CorruptSP()

$\mathfrak{C}_{\mathcal{SP}} := 1$

CorruptRA()

$\mathfrak{C}_{\mathcal{RA}} := 1$
return $\mathsf{sk}_{\mathcal{RA}}$

SubscribeQuery(qid)

$\mathsf{regQ}_{qid} \leftarrow \mathtt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$s \leftarrow \mathtt{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid)$
return s

ReportData($qid, m, \vec{s} := (s_1, \ldots, s_k)$)

$\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$c \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$
if $\mathfrak{C}_{\mathcal{SP}} = 1$ **then**
  return c
else
  for $i \in 1, \ldots, k$ **do**
    $c_i \leftarrow \mathtt{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, s_i)$
  endfor
  return $\vec{c} := (c_1, \ldots, c_k)$
endif

DecodeData(qid, c)

$\mathsf{regQ}_{qid} \leftarrow \mathtt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$m \leftarrow \mathtt{DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid, c)$
return m

Figure 4.2: Oracles available to the adversary in PEPSICo

4.2.1 Node privacy

The *node privacy* notion formalizes the confidentiality of the reported data and the query identity. It requires the reported data and the query identity to be hidden from the SP as well as unauthorized queriers and other mobile nodes. It is modeled as indistinguishability of data reports against an adaptive adversary which could be a collusion of multiple of the parties mentioned before. It is distinguished between *node privacy* under *chosen-plaintext attacks* and under *chosen-ciphertext attacks*. In the weaker first notion, the adversary has no access to the decoding oracle. This notion is important for PEPSICo instantiations that implement the `AggregateData` algorithm as such instantiations cannot achieve the stronger notion.

Definition 4.2 (Node privacy) *Let* PI *be a PEPSICo instantiation and* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *an* PPT *adversary, then the game* $\text{Game}^{\text{NP-CCA}}_{\text{PI},\mathcal{A}}(n)$ *is defined in Figure 4.3.*

$\text{Game}^{\text{NP-CCA}}_{\text{PI},\mathcal{A}}(n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{Setup}(1^n)$

$((qid_0, m_0), (qid_1, m_1), \vec{s}, state_0) \leftarrow \mathcal{A}_0^{\text{CorruptMN,CorruptQ,CorruptSP,ReportData,SubscribeQuery,DecodeData}}(\mathsf{pk}_{\mathcal{RA}})$

$b \xleftarrow{\$} \{0,1\}$

$\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid_b)$

$c \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid_b}, qid_b, m_b)$

if $\mathfrak{C}_{\mathcal{SP}} \stackrel{?}{=} 1$ **then**

  $b' \leftarrow \mathcal{A}_1^{\text{CorruptMN,CorruptQ,CorruptSP,ReportData,SubscribeQuery,DecodeData}}(\mathsf{pk}_{\mathcal{RA}}, c, state_0)$

else

  for $i \in \{0, \ldots, k\}$ **do**

    $c_i \leftarrow \mathtt{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, s_i)$

  endfor

  $b' \leftarrow \mathcal{A}_1^{\text{CorruptMN,CorruptQ,CorruptSP,ReportData,SubscribeQuery,DecodeData}}(\mathsf{pk}_{\mathcal{RA}}, \vec{c} = (c_0, \ldots, c_k), state_0)$

endif

return $b \stackrel{?}{=} b'$ and $\{qid_0, qid_1\} \cap (\mathfrak{CI}_{\text{MN}} \cup \mathfrak{CI}_{\mathcal{Q}}) \stackrel{?}{=} \emptyset$

Figure 4.3: NP-CCA game for PEPSICo

The adversary $\mathcal{A}$ *has the following additional restrictions:*

- $\mathcal{A}$ *must not query* SubscribeQuery *with* qid_0 *or* qid_1
- *If* $\mathfrak{C}_{\mathcal{SP}} \stackrel{?}{=} 1$*, then* $\mathcal{A}$ *is not allowed to query* ReportData *with* qid_0 *or* qid_1
- $\mathcal{A}_1$ *is not allowed to query* DecodeData *for* qid_0 *or* qid_1 *together with* c *(if* $\mathfrak{C}_{\mathcal{SP}} \stackrel{?}{=} 1$*) or any element of* $\vec{c}$ *(if* $\mathfrak{C}_{\mathcal{SP}} \neq 1$*)*

PI *is called* NP-CCA *secure if no adversary can win this game with more than negligible probability, ie. for all* PPT $\mathcal{A}$

$$\text{Adv}^{\text{NP-CCA}}_{\text{PI},\mathcal{A}}(n) := \left| \Pr\left[\text{Game}^{\text{NP-CCA}}\text{PI}, \mathcal{A}(n) = 1\right] - \frac{1}{2} \right| \leq \text{negl}(n)$$

Let the game $\text{Game}^{\text{NP-CPA}}_{\text{PI},\mathcal{A}}(n)$ *be identical to* $\text{Game}^{\text{NP-CCA}}_{\text{PI},\mathcal{A}}(n)$ *except that* $\mathcal{A}$ *is not given access to the* DecodeData *oracle and* $\text{Adv}^{\text{NP-CPA}}_{\text{PI},\mathcal{A}}(n)$ *be analogously defined as above.* PI *is called* NP-CPA *secure if* $\text{Adv}^{\text{NP-CPA}}_{\text{PI},\mathcal{A}}(n) \leq \text{negl}(n)$ *for all* PPT *adversaries* $\mathcal{A}$.

4.2.2 Query privacy

The *query privacy* property protects the privacy of the queriers by hiding the query identity of a subscription from the SP, MNs, and other queriers. The property is modeled as indistinguishability of subscription tokens for two query identities.

Definition 4.3 (Query privacy) *Let* PI *be a PEPSICo instantiation and* $\mathcal{A} = (\mathcal{A}_0), \mathcal{A}_1)$ *an* PPT *adversary, then the game* $\text{Game}^{\text{QP}}_{\text{PI},\mathcal{A}}(n)$ *is defined in Figure 4.4. In the game,* $\mathcal{A}$ *must not query* ReportData *or* SubscribeQuery *with* qid_0 *or* qid_1.

$\text{Game}^{\text{QP}}_{\text{PI},\mathcal{A}}(n)$

$(\text{pk}_{\mathcal{RA}}, \text{sk}_{\mathcal{RA}}) \leftarrow \texttt{Setup}(1^n)$

$(qid_0, qid_1, state_0) \leftarrow \mathcal{A}_0^{\text{CorruptMN,CorruptQ,ReportData,SubscribeQuery,DecodeData}}(\text{pk}_{\mathcal{RA}})$

$b \xleftarrow{\$} \{0,1\}$

$\text{regQ}_{qid_b} \leftarrow \texttt{RegisterQ}(\text{pk}_{\mathcal{RA}}, \text{sk}_{\mathcal{RA}}, qid_b)$

$s \leftarrow \texttt{SubscribeQuery}(\text{pk}_{\mathcal{RA}}, \text{regQ}_{qid_b}, qid_b)$

$b' \leftarrow \mathcal{A}_1^{\text{CorruptMN,CorruptQ,ReportData,SubscribeQuery,DecodeData}}(\text{pk}_{\mathcal{RA}}, s, state_0)$

return $b \stackrel{?}{=} b'$ and $\{qid_0, qid_1\} \cap (\mathfrak{CI}_{\mathcal{Q}} \cup \mathfrak{CI}_{\text{MN}}) \stackrel{?}{=} \emptyset$

Figure 4.4: *Query privacy* game for PEPSICo

PI *provides* query privacy *if all* PPT *adversaries win the game with at most negligible probability, ie for all* PPT $\mathcal{A}$

$$\text{Adv}^{\text{QP}}_{\text{PI},\mathcal{A}}(n) := \left| \Pr\left[\text{Game}^{\text{QP}}\text{PI}, \mathcal{A}(n) = 1\right] - \frac{1}{2} \right| \leq \text{negl}(n)$$

4.2.3 Report unlinkablility

To protect the privacy of users, PEPSICo demands that no other party should be able to link two data reports as being generated by the same user. They explicitly include the RA in their definition. For example, this prevents to trace the location of mobile nodes (if included in the sensing data) to generate profiles that might be used to identify the owner.

As MNs do not have identifiers in the model, the security definition models report unlinkability as indistinguishably of the mobile node registration value used to generate a report. The query identity and message are freely chosen by the adversary. Obviously, this property demands that more than one MN has to be registered for the query identity. Note that this property only holds with respect to an honest-but-curious RA as, even if the RA is corrupted, the mobile node registration values are generated by the experiment.

Definition 4.4 (Report unlinkability) *Let* PI *be a PEPSICo instantiation and* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *an* PPT *adversary, then the game* $\text{Game}^{\text{RU}}_{\text{PI},\mathcal{A}}(n)$ *is defined in Figure 4.5.*

$\text{Game}^{\text{RU}}_{\text{PI},\mathcal{A}}(n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \xleftarrow{\$} \mathtt{Setup}(1^n)$

$(qid, m, state_0) \leftarrow \mathcal{A}_0^{\text{CorruptMN,CorruptQ,CorruptRA,ReportData,SubscribeQuery,DecodeData}}(\mathsf{pk}_{\mathcal{RA}})$

$\mathsf{regMN}^0_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

$\mathsf{regMN}^1_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

$b \xleftarrow{\$} \{0,1\}$

$c \xleftarrow{\$} \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}^b_{qid}, qid, m)$

$b' \leftarrow \mathcal{A}_1^{\text{CorruptMN,CorruptQ,CorruptRA,ReportData,SubscribeQuery,DecodeData}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}^0_{qid}, \mathsf{regMN}^1_{qid}, c, state_0)$

return $b \stackrel{?}{=} b'$

Figure 4.5: *Report unlinkability* game for PEPSICo

A PEPSICo instanciation PI *provides* report unlinkability *if there exist a negligible function* negl *such that*

$$\text{Adv}^{\text{RU}}_{\text{PI},\mathcal{A}}(n) := \left| \Pr\left[\text{Game}^{\text{RU}}_{\text{PI},\mathcal{A}}(n) = 1\right] - \frac{1}{2} \right| \leq \text{negl}(n)$$

4.3 Discussion

While PEPSICo guarantees strong privacy and security properties even against collusion attacks, it still has some limitations. PEPSICo provides report unlinkability only with respect to malicious RA. However, this is a weak trust assumption compared with other participatory sensing models. Moreover, in the *report unlinkability* experiment, the mobile node registration values are generated after the adversary outputs the query identity and message pair, and, therefore, unknown to the adversary when computing those. Thus, PEPSICo does not offer forward privacy. Once the mobile node registration value of a user has been exposed, *report unlinkability* does not hold for subsequent transactions, even if the adversary has no further access to private information of the mobile node.

In addition, *node privacy* and *query privacy* do not hold against a collusion of the SP and a mobile node registered for the query identity in question. It depends on the openness of user registration, whether it might be difficult for an adversary controlling the SP to just register for each query identity to circumvent these properties. Moreover, side-channel attacks might be possible. As one of the design goals is to make it easy to register as a querier, it is likely that there will be queriers registering few query identities, if not only a single one. If the RA publishes the responsible queriers together with the query identity and task this could allow the SP to derive the query identity of data reports. Moreover, the querier might publish their results, from which the query identity could be derived. Depending on the system and business model, the SP could know which querier uses which subscription token as the connection could be identifying, eg. for billing purposes. Therefore, *node privacy* and *query privacy*

could be circumvented. However, in the instantiations they provide, the reported data will remain confidential even if the query identity of a report gets exposed.

Furthermore, the Network Operator (NO), which was considered in PEPSI, has been dropped from the system model because of its attack capabilities were strictly weaker than the SP. However, it should be mentioned that in PEPSI, *report unlinkability* does not hold with respect to the NO. This property is inherited by PEPSICo. Therefore, it is necessary to use an anonymizing network if the network operator is not trusted.

5 BBA+

BBA+ [Har+17] is a secure point collection and redemption system with a strong focus on user privacy. It allows users to anonymously collect and redeem points while preventing cheating. It is an extension of Black-box Accumulation (BBA) [JR16] but has stronger security properties and supports some additional features. Because of its privacy properties, BBA+ is highly suitable to be used as an incentive mechanism in participatory sensing.

In this chapter, we describe the system model of BBA+ and provide its formal system definition and security properties.

5.1 The BBA+ model

The BBA+ model consists of the following parties:

Trusted Third Party (TTP)
: The TTP is only required once in the setup phase of the scheme to generate the CRS that describes the algebraic framework and contains system-wide public keys.

Accumulator
: The accumulator interacts with users to add points to their balance.

Verifier
: The verifier interacts with the users to enable them to verify that they possess the claimed balance and to redeem points.

Issuer
: The issuer interacts with the users so that they can obtain their initial balance tokens. Thereby, it verifies that a user is in possession of their secret key and that their public key is indeed unique.

User
: The participants of the system that may collect and redeem points.

The accumulator, verifier, and issuer are called operators. They are required to trust each other as they share the same secret key.

Figure 5.1 outlines the working of the model. To participate, a user generates a public and private key pair invokes the issue protocol with the issuer to obtain an initial balance token with balance 0. To add points to the token's balance, the user runs the accumulation protocol with the accumulator

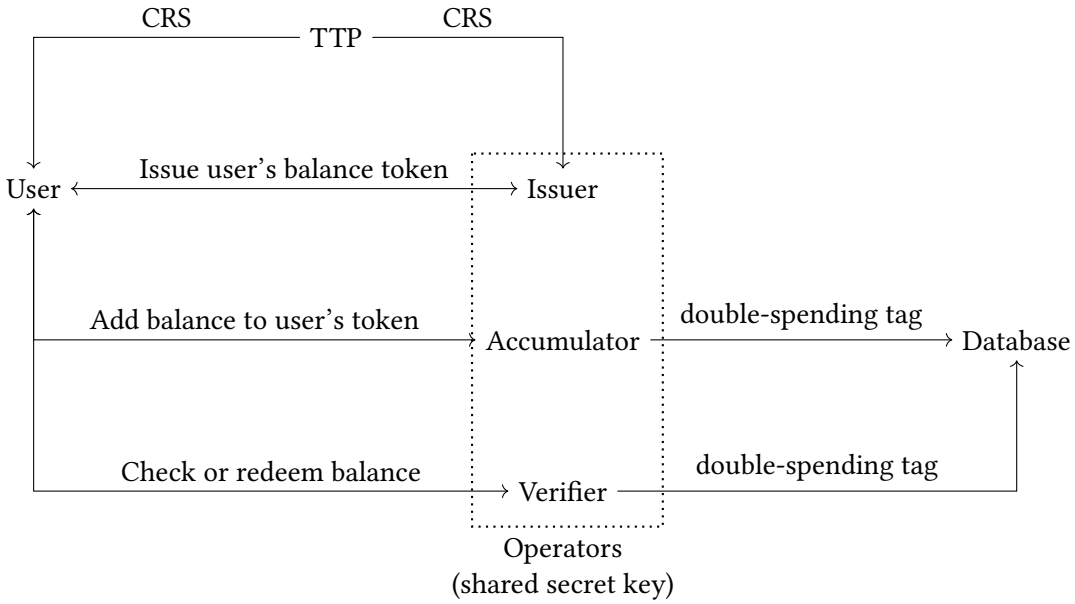

Figure 5.1: Entities in the BBA+ model

and obtains an updated token with the modified balance. Hereby, the added value might be positive or negative. The accumulator outputs a double-spending tag which allows identifying users that cheat by using old tokens. Those double-spending tags are stored within a global database, where regular checks are applied. The verification protocol allows interacting with the verifier, either to prove that the user's token has a certain balance or to redeem points from this balance. This protocol is very similar to the accumulation protocol. The user obtains an updated token and the verifier a double-spending tag.

The interesting part is that the users are anonymous during the accumulation and verification protocol and that transactions are unlikable.

Definition 5.1 (BBA+) *An BBA+ scheme consists of a tuple of algorithm and interactive protocols (*`Setup`*,* `IGen`*,* `UGen`*,* `Issue`*,* `Accum`*,* `Verify`*,* `UVer`*,* `IdentDS`*,* `VerifyGuilt`*).*

$(\mathsf{CRS}, \mathsf{td}) \xleftarrow{\$} \mathtt{Setup}(1^n)$

The setup algorithm has to be executed by a TTP. It returns the public common reference string CRS *and a trapdoor* td*. The trapdoor is only required to define the security notions and has to be kept secret.*

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \xleftarrow{\$} \mathtt{IGen}(\mathsf{CRS})$

With this algorithm, the issuer generates his public and secret key, which has to be shared with the accumulator and verifier as well. For convenience, it is assumed that CRS is part of $\mathsf{pk}_{\mathcal{I}}$*.*

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \xleftarrow{\$} \mathtt{UGen}(\mathsf{CRS})$

With this algorithm, a user generates his personal public/private key pair.

$((\tau, b_{\mathcal{U}}), b_{\mathcal{I}}) \xleftarrow{\$} \mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$

The interactive token issuing protocol is executed between a user $\mathcal{U}$ and the issuer $\mathcal{I}$ and results in the user outputing a balance token τ with a balance of 0. The bits $b_{\mathcal{U}}$ and $b_{\mathcal{I}}$ indicate whether the user and the issuer accept the protocol run.

$((\tau^*, b_{\mathcal{U}}), (\mathsf{dstag}, \mathsf{hid}, b_{\mathcal{AC}})) \xleftarrow{\$} \mathtt{Accum}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v), \mathcal{AC}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v)\right\rangle$

The interactive accumulation protocol is executed between a user $\mathcal{U}$ and the accumulator $\mathcal{AC}$. τ is the user's balance token with balance w and v the value that should be added to the balance. τ^* is a new token with balance $w + v$. The issuer outputs a double-spending tag $\mathsf{dstag} = (s,z)$ with token version number s and data z and a hidden user ID hid, which is for definitorial purpose only. The bits $b_{\mathcal{U}}$ and $b_{\mathcal{AC}}$ indicate whether the parties accept the protocol run.

$((\tau^*, b_{\mathcal{U}}), (\mathsf{dstag}, \mathsf{hid}, b_{\mathcal{V}})) \xleftarrow{\$} \mathtt{Verify}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v), \mathcal{V}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v)\right\rangle$

The interactive verification and redeeming protocol is analogous to the accumulation protocol, with the accumulator exchanged with the verifier, which gets the current token balance w as additional input.

$b \leftarrow \mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w)$

With this algorithm, a user can check whether τ is indeed a valid token with the balance w that is owned by him.

$(\mathsf{pk}_{\mathcal{U}}, \Pi)$ or $\perp \leftarrow \mathtt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_1, \mathsf{dstag}_2)$

The double-spender detection algorithm checks whether two double-spending tags were generated by reusing the same token. If this has been the case, it outputs the public key $pk_{\mathcal{U}}$ of the according user together with a proof of guilt Π, else it returns $\perp$.

$b \leftarrow \mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi)$

This algorithm can be used to check if a user is guilty of double-spending. It outputs 1 is Π proofs that the user with the public key $\mathsf{pk}_{\mathcal{U}}$ is guilty of double-spending.

A BBA+ scheme is called correct *if the following properties hold for all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$, issuer key-pairs $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$, user key-pairs $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$ and parties $\mathcal{U}$, $\mathcal{I}$, $\mathcal{AC}$ and $\mathcal{V}$ honestly following the protocols:*

Correctness of issuing

For all outputs of the issue protocol $((\tau, b_{\mathcal{U}}), b_{\mathcal{I}}) \xleftarrow{\$} \mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$, it holds that

$$b_{\mathcal{U}} \stackrel{?}{=} b_{\mathcal{I}} \stackrel{?}{=} 1 \quad \wedge \quad \mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, 0) \stackrel{?}{=} 1$$

Correctness of accumulation

For all tokens τ, balances $w \in \mathbb{Z}_p$ with $\mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w) \stackrel{?}{=} 1$ and all values $v \in \mathbb{Z}_p$, we

have that

$$((\tau^*, 1), (\text{dstag}, \text{hid}, 1)) \leftarrow \texttt{Accum}\left\langle \mathcal{U}(\text{pk}_\mathcal{I}, \text{pk}_\mathcal{U}, \text{sk}_\mathcal{U}, \tau, w, v), \mathcal{AC}(\text{pk}_\mathcal{I}, \text{sk}_\mathcal{I}, v)\right\rangle$$
$$\wedge\ \texttt{UVer}(\text{pk}_\mathcal{I}, \text{pk}_\mathcal{U}, \text{sk}_\mathcal{U}, \tau^*, w + v) \stackrel{?}{=} 1$$

Correctness of token verification

For all tokens τ, balances $w \in \mathbb{Z}_p$ with $\texttt{UVer}(\text{pk}_\mathcal{I}, \text{pk}_\mathcal{U}, \text{sk}_\mathcal{U}, \tau, w) \stackrel{?}{=} 1$ and values $v \in \mathbb{Z}_p$, we have that

$$((\tau^*, 1), (\text{dstag}, \text{hid}, 1)) \stackrel{\$}{\leftarrow} \texttt{Verify}\left\langle \mathcal{U}(\text{pk}_\mathcal{I}, \text{pk}_\mathcal{U}, \text{sk}_\mathcal{U}, \tau, w, v), \mathcal{V}(\text{pk}_\mathcal{I}, \text{sk}_\mathcal{I}, w, v)\right\rangle$$
$$\wedge\ \texttt{UVer}(\text{pk}_\mathcal{I}, \text{pk}_\mathcal{U}, \text{sk}_\mathcal{U}, \tau^*, w + v) \stackrel{?}{=} 1$$

5.2 Security of BBA+

BBA+ categorizes its security properties into two categories: System security which protects the system from cheating users and user security and privacy, which protects the user from a cheating or tracking system. In the following, the security properties within these categories are described.

5.2.1 System security

From the system operator's point of view, we want to have three properties enforced. Balance tokens should be *owner-binding*, meaning that they can only be used by their legitimate owner, *balance-binding*, meaning that there is no way to claim a false balance, and we want to prevent users from using old tokens. BBA+ achieves this by ensuring that users presenting old tokens can be identified later (*double-spending detection*). If all protocol messages are additionally encrypted with an IND-CCA secure encryption scheme, BBA+ is additionally secure against eavesdropping, which is not considered in the following security definitions.

For the definitions, we define the oracles MalIssue, MalAcc and MalVer as in Figure 5.2. Moreover, $\mathcal{V}^{\texttt{Accum}}_{n,\text{CRS}}$ denotes the set of the accumulators views of the `Accum` protocol runs for a fixed security parameter n and CRS CRS, consisting of all its inputs, outputs and messages send and received, i.e., $(\text{pk}_\mathcal{I}, \text{sk}_\mathcal{I}, v, \text{msgs}, \text{dstag}, \text{hid}, b_{\mathcal{AC}})$, where $msgs \in \{0,1\}^*$ is the bit string of all sent messages. Analogously, $\mathcal{V}^{\texttt{Verify}}_{n,\text{CRS}}$ defines the set of views of the verifier on the `Verify` protocol runs.

Trapdoor-linkability

To be able to define the required security properties, we need to be able to link each transaction with a user and token. However, we demand these transactions to be anonymous and unlinkable. This conflict is resolved by introducing a trapdoor that allows abolishing privacy. This trapdoor has to be kept secret by the TTP.

MalIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathfrak{M} \cap \mathsf{pk}_{\mathcal{U}} \stackrel{?}{=} \emptyset$ **then**
  $\mathfrak{M} \leftarrow \mathfrak{M} \cup \mathsf{pk}_{\mathcal{U}}$
  $\texttt{Issue}\left\langle \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$
endif

MalAcc(v)

$\texttt{Accum}\left\langle \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v), \mathcal{AC}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v)\right\rangle$

MalVer(w, v)

$\texttt{Verify}\left\langle \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{U}}, w_{\mathcal{U}}, v), \mathcal{V}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v)\right\rangle$

Figure 5.2: Oracle definitions for the BBA+ system security properties. The outputs of the interactive protocols have been omitted in the figure.

Definition 5.2 (Trapdoor-linkability) *A BBA+ scheme* BBA+ *is called* trapdoor-linkable *if it satisfies the following tow conditions:*

Completeness

For all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \texttt{Setup}(1^n)$ *and* $\mathsf{view} = (\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v, \mathsf{msgs}, \mathsf{dstag}, \mathsf{hid}, 1) \in \mathcal{V}^{\texttt{Accum}}_{n,\mathsf{CRS}} \cup \mathcal{V}^{\texttt{Verify}}_{n,\mathsf{CRS}}$ *there exist inputs* $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w)$ *and random choices for an honest user* $\mathcal{U}$ *and an honest accumulator* $\mathcal{AC}$ *such that if they run the respective protocol this leads to a view* view' *containing the same hidden user ID* hid *as in* view.

Extractability

For all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \texttt{Setup}(1^n)$ *and* $\mathsf{view} = (\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v, \mathsf{msgs}, \mathsf{dstag}, \mathsf{hid}, 1) \in \mathcal{V}^{\texttt{Accum}}_{n,\mathsf{CRS}} \cup \mathcal{V}^{\texttt{Verify}}_{n,\mathsf{CRS}}$ *resulting from a protocol run with an honest user on input* $\mathsf{pk}_{\mathcal{U}}$*, there exist a* PPT *algorithm* ExtractUID *such that*

$$\mathsf{pk}_{\mathcal{U}} \leftarrow \texttt{ExtractUID}(\mathsf{td}, \mathsf{hid})$$

Owner-binding

This property ensures that a token can only be issued to its legitimate owner and cannot be used by anyone but its legitimate owner.

Definition 5.3 (Owner-binding) *A trapdoor-linkable BBA+ scheme* BBA+ *is called* owner-binding *if no* PPT *adversary can win* $\mathrm{Exp}^{\text{OB-issue}}_{\text{BBA+},\mathcal{A}}(n)$ *or* $\mathrm{Exp}^{\text{OB-acc-ver}}_{\text{BBA+},\mathcal{A}}(n)$ *as defined in Figure 5.3 with more than negligible probability. More precisely, for all* PPT *adversaries* $\mathcal{A}$

$$\Pr\left[\mathrm{Exp}^{\text{OB-issue}}_{\text{BBA+},\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n) \quad \text{and} \quad \Pr\left[\mathrm{Exp}^{\text{OB-acc-ver}}\text{BBA+}, \mathcal{A}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

Balance-binding

Only the exact amount of points that have legitimately been collected up to this point can be claimed for a token unless an old version of the token is present (this case is addressed by the *double-spending detection* property).

$\mathrm{Exp}^{\text{OB-issue}}_{\text{BBA+},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\text{MalIssue,MalAcc,MalVer}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalIssue on input of the given public key $\mathsf{pk}_{\mathcal{U}}$

$\mathrm{Exp}^{\text{OB-acc-ver}}_{\text{BBA+},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\text{MalIssue,MalAcc,MalVer}}(\mathsf{pk}_{\mathcal{I}})$
The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalAcc or MalVer such that `ExtractUID` applied to hid being part of the view of this call outputs a public key $\mathsf{pk}_{\mathcal{U}}$ for which there has been no seccessful execution of MalIssue up to this call.

Figure 5.3: *Owner-binding* experiments for the `Issue`, `Accum` and `Verify` protocols of BBA+

Definition 5.4 (Balance-binding) *A trapdoor-linkable BBA+ scheme BBAP is called* balance-binding *if for all* PPT *adversaries* $\mathcal{A}$ *and the experiment* $\mathrm{Exp}^{\text{BB}}_{\text{BBA+},\mathcal{A}}(1^n)$ *as defined in Figure 5.4*

$$\Pr\left[\mathrm{Exp}^{\text{BB}}_{\text{BBA+},\mathcal{A}}(1^n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

Double-spending detection

The BBA+ system cannot prevent users from using old tokens, which could potentially have a higher balance than the current token. However, this property ensures that the use of old tokens can be detected and the cheating user can be identified.

Definition 5.5 (Double-spending detection) *A trapdoor-linkable BBA+ scheme* BBA+ *ensures* double-spending detection *if for all* PPT *adversaries* $\mathcal{A}$ *and the experiment* $\mathrm{Exp}^{\text{DSD}}_{\text{BBA+},\mathcal{A}}$ *as defined in Figure 5.5*

$$\Pr\left[\mathrm{Exp}^{\text{DSD}}_{\text{BBA+},\mathcal{A}}(1^n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

5.2.2 User security and privacy

There are two properties within this category. From the user's point of view, we need to ensure that the system operator does not learn more than necessary and especially cannot track the user's transactions (privacy). The second property prevents the misuse of the double-spending detection mechanism by the system operator, demanding that users cannot be proven guilty of double-spending unless they are.

$\mathrm{Exp}^{\mathrm{BB}}_{\mathrm{BBA+},\mathcal{A}}(1^n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(n)$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\mathrm{MalIssue,MalAcc,MalVer}}(\mathsf{pk}_{\mathcal{I}})$

The experiment returns 1 iff $\mathcal{A}$ did successful call MalVer resulting in a view $\mathsf{view} = (\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v, \mathsf{msgs}, \mathsf{dstag}, \mathsf{hid}, 1) \in \mathcal{V}^{\mathtt{Verify}}_{n,\mathrm{CRS}}$ and extracted user public key $\mathsf{pk}_{\mathcal{U}} \leftarrow \mathtt{ExtractUID}(\mathsf{td}, \mathsf{hid})$ such that the following conditions are satisfied:

- all successful MalIssue/MalAcc calls produced unique token version numbers
- the claimed balance $w \in \mathbb{Z}_p$ does not equal the sum of previously collected accumulation values v for $\mathsf{pk}_{\mathcal{U}}$, i.e.,

$$w \neq \sum_{v \in \mathfrak{V}_{\mathsf{pk}_{\mathcal{U}}}} v,$$

  where $\mathfrak{V}_{\mathsf{pk}_{\mathcal{U}}}$ is the list of all accumulation values $v \in \mathbb{Z}_p$ that appeared in previous successful calls to MalAcc or MalVer for which $\mathsf{pk}_{\mathcal{U}}$ could be extracted using `ExtractUID`.

Figure 5.4: *Balance-binding* experiment for BBA+

$\mathrm{Exp}^{\mathrm{DSD}}_{\mathrm{BBA+},\mathcal{A}}$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}n$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\mathrm{MalIssue,MalAcc,MalVer}}(\mathsf{pk}_{\mathcal{I}})$

The experiment returns 1 iff $\mathcal{A}$ did two successful MalAcc/MalVer calls resulting in two views view_0 and view_1 including two double-spending tags $\mathsf{dstag}_0 = (s, z_0)$ and $\mathsf{dstag}_1 = (s, z_1)$ and extracted user public keys $\mathsf{pk}^{(0)}_{\mathcal{U}}$ and $\mathsf{pk}^{(1)}_{\mathcal{U}}$ (using `ExtractUID`) such that at least one of the following conditions is satisfied:

- $\mathsf{pk}^{(0)}_{\mathcal{U}} \neq \mathsf{pk}^{(1)}_{\mathcal{U}}$ or
- $\mathtt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1) \neq (\mathsf{pk}^{(0)}_{\mathcal{U}}, \Pi)$ or
- $\mathtt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1) \stackrel{?}{=} (\mathsf{pk}^{(0)}_{\mathcal{U}}, \Pi)$ but $\mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}^{(0)}_{\mathcal{U}}, \Pi) \stackrel{?}{=} 0$

Figure 5.5: *Double-spending detection* experiment for BBA+

Privacy

This property protects the user from tracking through the system operator (a potential collusion of $\mathcal{I}$, $\mathcal{AC}$ and $\mathcal{V}$). The operator may not be able to link `Accum` and `Verify` transactions of honest users, even for transactions preceding and succeeding (except for the very next) a corruption of the user (*forward and backward privacy*).

This is property is defined using the real/ideal paradigm. The adversary plays the role of the issuer, accumulator, and verifier and is allowed to interact with a couple of oracles that allow him to create and corrupt users as well as to interact with honest users. There is the restriction that whenever an interaction does not terminate successfully, the according user refuses to participate in any future interactions. In addition, concurrent oracle calls for the same user (same $\mathsf{pk}_{\mathcal{U}}$) are not allowed.

In the real world, the oracles behave according to the protocol, whereas in the ideal world, the honest users are substituted with a simulator that in most cases has no access to any user-related data. However, the simulator has to be able to provide a secret key, a plausible token and the correct balance for a user in case of his corruption. Therefore, the simulation must keep track of all balances. In case the adversary wants to interact with a user that has previously been corrupted, the simulator executes the real protocol with the information previously returned to the adversary instead of simulating its execution.

Definition 5.6 (Privacy) *A BBA+ scheme* BBA+ *is called* privacy-preserving*, if there exist* PPT *algorithms* `SimSetup` *and* SimCorrupt *and interactive* PPT *oracles* SimHonIssue, SimHonAcc *and* SimHonVer *as described in Figure 5.6, respectively, such that for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *and for the experiments defined in Figure 5.7*

$$\left|\Pr\left[\mathrm{Exp}^{\text{PRIV-real}}_{\text{BBA+},\mathcal{A}}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\text{PRIV-ideal}}_{\text{BBA+},\mathcal{A}}(n) \stackrel{?}{=} 1\right]\right| \leq \mathsf{negl}(n)$$

False accusation protection

No system operator should be able to forge a proof that a user has allegedly committed double-spending.

Definition 5.7 (False accusation protection) *A trapdoor-linkable BBA+ scheme* BBA+ *ensures* false-accusation protection*, if for all* PPT *adversary* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *and the experiment* $\mathrm{Exp}^{\text{FACP}}_{\text{BBA+},\mathcal{A}}$ *as defined in Figure 5.8 and the oracles defined in Figure 5.6*

$$\Pr\left[\mathrm{Exp}^{\text{FACP}}_{\text{BBA+},\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

HonUser()

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
return $\mathsf{pk}_{\mathcal{U}}$

RealHonIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \stackrel{?}{=} \emptyset$ **then**
  $\mathfrak{I} \leftarrow \mathfrak{I} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  $\mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$
endif

RealHonAcc($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ **then**
  $\mathtt{Accum}\left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{U}}, w_{\mathcal{U}}, v), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v) \end{matrix} \right\rangle$
endif

RealHonVer($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ **then**
  $\mathtt{Verify}\left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{U}}, w_{\mathcal{U}}, v), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v) \end{matrix} \right\rangle$
endif

RealCorrupt($\mathsf{pk}_{\mathcal{U}}$)

return $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{U}})$

HonUser()

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
return $\mathsf{pk}_{\mathcal{U}}$

SimHonIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \stackrel{?}{=} \emptyset$ **then**
  $\mathfrak{I} \leftarrow \mathfrak{I} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  $\mathtt{Issue}\left\langle \mathsf{Sim}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$
endif

SimHonAcc($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ **then**
  $\mathtt{Accum}\left\langle \mathsf{Sim}(v), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, v)\right\rangle$
endif

SimHonVer($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ **then**
  $\mathtt{Verify}\left\langle \mathsf{Sim}(w_{\mathcal{U}}, v), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v)\right\rangle$
endif

SimCorrupt($\mathsf{pk}_{\mathcal{U}}$)

return $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{U}}) \leftarrow \mathsf{Sim}(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, s)$

Figure 5.6: Oracles for the BBA+ user privacy experiments. For simplicity, the return values of the interactive protocols have been omitted. If a protocol run for a user identity is not accepted $b_{\mathcal{U}} = 0$ then the user will reject to further participate, ie. $\mathcal{A}$ cannot make any more oracle calls for this user identity.

$\underline{\mathrm{Exp}^{\text{PRIV-real}}_{\text{BBA+},\mathcal{A}}(n)}$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, state_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\text{HonUser,RealHonIssue,RealHonAcc,RealHonVer,RealCorrupt}}(\mathsf{pk}_{\mathcal{I}}, state_0)$
return b

$\underline{\mathrm{Exp}^{\text{PRIV-ideal}}_{\text{BBA+},\mathcal{A}}(n)}$

$(\mathsf{CRS}, \mathsf{td}_{sim}) \leftarrow \mathtt{SimSetup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, state_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\text{HonUser,SimHonIssue,SimHonAcc,SimHonVer,SimCorrupt}}(\mathsf{pk}_{\mathcal{I}}, state_0)$
return b

Figure 5.7: Real and ideal world user privacy experiments for BBA+

$\underline{\mathrm{Exp}^{\text{FACP}}_{\text{BBA+},\mathcal{A}}(n)}$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, state_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$
$\Pi \leftarrow \mathcal{A}_1^{\text{RealHonIssue,RealHonAcc,RealHonVer}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, state_0)$
return $\mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi) \stackrel{?}{=} 1$

Figure 5.8: *False accusation protection* experiment for BBA+

6 Interim Model

In the following, we present an interim model that extends PEPSICo by using BBA+ as an incentive mechanism. First, we discuss the design choices and provide an informal overview of the model before the formal definition of the algorithms and protocols is given and the security requirements are looked at. Thereafter, we give an implementation of our model and prove that it satisfies the specified requirements. Last, we identify several shortcomings in this model, leading to the advanced model presented in Chapter 7.

6.1 Discussion on design choices

The idea behind the interim model is to combine PEPSICo and BBA+ as straight forward as possible to create a participatory sensing model with an incentive mechanism. This allows using these models mainly as a black box and to reuse most of their security properties. We use this model as an interim step towards our final model, allowing to reflect on its strengths and weaknesses.

The main decisions for the interim model are the design of the incentive mechanism and which entities are required within the model. We discuss those choices within the following sections.

6.1.1 Incentive mechanism

The first important design choice is how the incentive mechanism shall look like. In this regard, our scheme features a central incentive mechanism instead of handling incentives on a per querier basis. We argue that it is considerably less effort to register as a querier if you do not have to provide own infrastructure to reward the users. Moreover, it is also more convenient from the user's perspective if incentives gathered from multiple queriers are accumulated into a single account. Therefore, incentives should be handled by the system and not by individual queriers. The problem is that the system has no way to determine the value of the provided information as we want the report type (query identity) as well as the data to be only known to queriers that have registered for the specific query identity. Therefore, they are the only parties that can determine the quality and value of the reported data and thus can decide on the number of incentives that should be rewarded for providing that data. This is also the reason why we will not include report aggregation, as one bad data report could render the aggregate useless.

The querier, therefore, has to report the value of the data, corresponding to the number of incentives the mobile node should receive, to the system where it can be collected by the mobile node. However,

this requires linkability between the report and the mobile node. Moreover, if it should be possible to sort out bad data directly at the SP, the querier could additionally report the quality of the data (or the incentives could be distributed based on the quality), which would make the complete transaction traceable through the system. Despite this, it has to remain impossible to link multiple transactions as originating from the same user.

An alternative possibility would be to reward each report with a fixed number of incentives. A scheme for this could be obtained by using BBA+ between the SP and the MNs, without changing the PEPSICo scheme. Sending a report could be combined with the accumulation protocol to obtain the incentives. However, this approach has a major drawback. So far, we haven't considered users presenting wrong information. However, using this alternative, a user could just repeatedly report random data to collect arbitrary many incentives as the SP cannot even check the plausibility of the data. To address this problem, feedback from the queriers is required in any case. This is why restricting the model to fixed incentive rewards would not result in a considerable reduction in complexity in the long run. However, we are not completely addressing the problem of malicious data reports within the interim model.

6.1.2 Entities in the interim model

In this section, we discuss which entities are used within the interim model. As our model combines PEPSICo with BBA+, we start with the entities of the individual schemes and discuss which should be combined and which should be kept separated.

First, the MNs from PEPSICo and the users from BBA+ have to be combined as they both represent the sensor device (and its owner). Moreover, we want to reward users with the incentives, therefore they require an incentive account, corresponding to a user in BBA+.

Second, we keep the queriers as separate entities. As discussed in the previous section, our model implements a central incentive system independent of the querier. However, the querier is responsible to determine the quality of the received data and the number of incentives that the users should obtain.

To simplify the model, we can combine the issuer, accumulator and verifier from the BBA+ scheme, as they have to fully trust each other anyhow. We call the combined entity ISP (denoted with $\mathcal{I}$) and keep it as a separate entity from the SP to remain more flexible. However, as they are both part of the untrusted infrastructure, they could be combined in the actual implementation.

Last, we cannot combine the TTP with the RA. Even though the RA is partially trusted, eg. not to violate the *node privacy* of PEPSICo, *report unlinkability* should hold against the RA. As the different parts of a transaction have to be linkable (at least by a collusion of SP, querier and ISP) to allow the number of incentives being determined based on the reported data, the trapdoor of the BBA+ scheme would allow the TTP to link transactions to users.

While the users and queriers are offline entities and therefore only require a network connection whenever they want to send or receive data or register for a new query, the SP, and ISP, as well as the RA, are online entities that have to be constantly available.

6.2 Overview of the model

This section outlines the architecture and working of the interim model that has been derived from the previously discussed design choices.

The interim model consists of two phases, a setup phase that has to be performed once in the beginning and an operation phase. In the setup phase, the public parameters describing the scheme are established together with the secrets of the RA and ISP. Within the operation phase, mobile nodes and queriers can register for query identities as well as send and collect data reports and incentives.

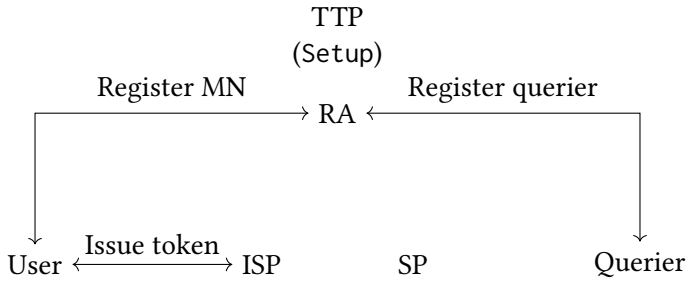

Figure 6.1: Setup and registration in the interim model

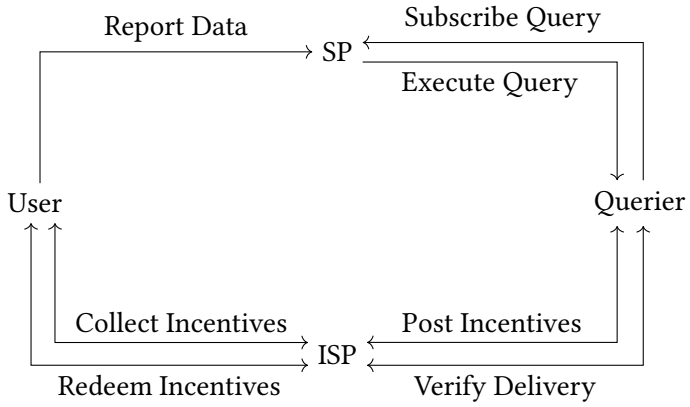

Figure 6.2: Operations of the interim model

The model is outlined within Figure 6.1 and Figure 6.2 with just one user and querier. The first figure shows the setup phase as well as the registration process for a querier and a user.

First, the TTP is required for the setup of a common reference string for the incentive scheme. Then, the RA and the ISP can generate their key pairs and publish their public keys. New users can generate a key pair at any time and have a balance token issued by the ISP.

The RA should also publish a list of available query identities together with a description of the gathered data. A querier could either register for an existing query identity, where special conditions/authentication may apply, or for a new query identity where he could define what data should be gathered. Moreover, users could look at the list with the queries and register with the RA for those they

are able and willing to provide data for.

After registration, queriers can subscribe to receive reports for query identities at the SP and users can generate data reports and send them to the service provider.

To match incentives to users, a bulletin mechanism is used. When creating a data report, the user also creates a bulletin nonce and secret and sends the nonce to the SP together with the report. The SP relays it to the querier and after evaluating the reported data, he can determine the appropriate number of incentives and send this information to the ISP. The ISP posts the nonce and the incentive value on a virtual bulletin board, allowing anyone with knowledge of the respective secret to claim the incentives, which is then be accumulated to the users' balance token. At any time, a user can redeem all or some of his accumulated incentives at the ISP.

6.3 Formal definition

Definition 6.1 (Interim I3PS) *An interim I3PS scheme consists of the following algorithms:*

$(\mathsf{CRS}, \mathsf{td}) \xleftarrow{\$} \mathtt{Setup}(1^n)$

The Setup algorithm for the incentive scheme is executed by the TTP once before the scheme can be used. The common reference string CRS *is made public while the trapdoor* td *remains a secret only used to define the security of the scheme.*

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \xleftarrow{\$} \mathtt{SetupRA}(1^n)$

With this algorithm, the RA creates its public and private key pair. The private key allows the RA to generate registration tokens for users and queriers.

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

This algorithm is executed by the ISP once to generate its public and private key pair. For convenience, it is assumed that CRS *is part of* $\mathsf{pk}_{\mathcal{I}}$.

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$

Every user is required to generate a public and private key pair for the incentives scheme. The public key acts as an identifier for the user.

$((\tau, b_{\mathcal{U}}), b_{\mathcal{I}}) \leftarrow \mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \right\rangle$

The next step for a user would be to request a balance token. This is done by engaging in the issue protocol with the ISP. During this protocol, the identity of the user is exposed.

$\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

This algorithm is executed by the RA to generate a registration value for a query identity which enables a user to submit data reports. As in PEPSICo, this algorithm is completely independent of the registering user's identity.

$(c, p, r) \leftarrow \texttt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$

With this algorithm, a user can, given his report data m and a registration value regMN_{qid} generate a data report for a query identity qid. In addition to the data report c, the algorithm outputs a bulletin nonce p and an corresponding secret r. The secret allows the user to claim any incentives posted for the corresponding bulletin nonce.

$((\tau', b_{\mathcal{U}}, w'), (\mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Collect}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, r), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, p, v)\right\rangle$

With this algorithm, a user with a balance token τ with balance w and a bulletin secret r can request any incentives posted for the corresponding nonce. After the protocol, the user has an updated token τ' with the balance w' that should be equal to $w + v$, whereby v is the sum of the incentives currently posted for the bulletin nonce p. Furthermore, the ISP obtains a double spending tag dstag *which can be used to detect cheating users. The hidden user id* hid *is only required for definitional purpose.*

$((\tau', b_{\mathcal{U}}, w'), (\mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Redeem}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v)\right\rangle$

This algorithm allows a user to redeem a specific number of incentives from his tokens balance. This algorithm exposes the balance of the token to the ISP to allow verifying that it is sufficient for retrieving the specified number of incentives. After the protocol, the user has an updated token τ' with the balance w' that should be equal to $w + v$ (where $v < 0$). Again, the ISP obtains a double spending tag dstag *to identify cheating users and* hid *is for the definition only.*

$b \leftarrow \texttt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w)$

This algorithm allows a user to verify that his balance token τ is indeed a valid balance token with the balance w.

$\mathsf{regQ}_{qid} \leftarrow \texttt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

With this algorithm, the RA computes a querier's registration value for a given query identity qid.

$s \leftarrow \texttt{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid)$

The query subscription algorithm allows an querier to compute a subscription token s for a query identity qid if he is in the possession of valid registration value for qid.

(c, p) or $\perp \leftarrow \texttt{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, p, s)$

This algorithm allows the SP to determine whether a report c matches a subscription toke s. In case of success, the report c and the bulletin nonce p have to be sent to the corresponding querier.

m or $\perp \leftarrow \texttt{DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid, c)$

The decoding algorithm is used by queriers to obtain the data m from a report c for a query identity qid for which he has a registration value regQ.

$(\mathsf{pk}_{\mathcal{U}}, \Pi)$ or $\perp \leftarrow \texttt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1)$

Given two double spending tags dstag_0 *and* dstag_1*, this algorithm outputs the public key of a user if the double spending tags originate from this user using the same balance token twice to collect or*

redeem incentives. Moreover, it outputs a proof Π that allows verifying that the owner of $\mathsf{pk}_{\mathcal{U}}$ is indeed guilty of double spending.

$b \leftarrow \mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi)$

This algorithm allows verifying a proof Π to see if the user identified by $\mathsf{pk}_{\mathcal{U}}$ is guilty of double spending.

An interim I3PS scheme is called correct *if the following properties hold for all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$, $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$, ISP key-pairs $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$, user key-pairs $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGenCRS}$ and parties $\mathcal{U}$ and $\mathcal{I}$ honestly following the protocols*

Correctness of issuing

For all outputs of the issue protocol $((\tau, b_{\mathcal{U}}), b_{\mathcal{I}}) \xleftarrow{\$} \mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \right\rangle$, it holds that

$$b_{\mathcal{U}} \stackrel{?}{=} b_{\mathcal{I}} \stackrel{?}{=} 1 \quad \wedge \quad \mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, 0) \stackrel{?}{=} 1$$

Correctness of data reporting

For all sensing data $m \in \mathfrak{M}$, query identities $qid \in \mathfrak{I}$, mobile node registration values $\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$, querier registration values $\mathsf{regQ}_{qid} \leftarrow \mathtt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$, subscription tokens $s \leftarrow \mathtt{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$ and $(c, p, r) \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$, we have that

$$\begin{aligned} &\mathtt{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, p, s) \stackrel{?}{=} (c, p) \\ \wedge\ &\mathtt{DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, c) \stackrel{?}{=} m \end{aligned}$$

Correctness of collection

For all tokens τ, balances $w \in \mathbb{Z}_p$ with $\mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w) \stackrel{?}{=} 1$, values $v \in \mathbb{Z}_p$, sensing data $m \in \mathfrak{M}$, query identities $qid \in \mathfrak{I}$, mobile node registration values $\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$ and $(c, p, r) \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$, we have that

$$\begin{aligned} &((\tau^*, 1, w'), (\mathsf{dstag}, \mathsf{hid}, 1)) \leftarrow \mathtt{Collect}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, r), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, p, v) \right\rangle \\ \wedge\ &\mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*, w') \stackrel{?}{=} 1 \\ \wedge\ &w' \stackrel{?}{=} w + v \end{aligned}$$

Correctness of redemption

For all tokens τ, balances $w \in \mathbb{Z}_p$ with $\mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w) \stackrel{?}{=} 1$ and values $v \in \mathbb{Z}_p$, we have that

$$\begin{aligned} &((\tau^*, 1), (\mathsf{dstag}, \mathsf{hid}, 1)) \xleftarrow{\$} \mathtt{Redeem}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, v), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v) \right\rangle \\ \wedge\ &\mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*, w + v) \stackrel{?}{=} 1 \end{aligned}$$

6.4 Security

In this section, we first discuss informally what security properties are needed for the extended model and which of them can be directly carried over from PEPSICo and BBA+. Second, we provide formal definitions for the new or adapted properties.

6.4.1 Requirements

Our model has three fundamental security goals. First, we want to protect the privacy of users and prevent tracking based on the transmitted data reports. Second, the confidentiality of the transmitted data itself should be guaranteed and third, the incentive system should be secure and prevent cheating.

Unlinkability of transactions

To prevent the profiling of users, no other party (nor a collusion of other parties) should be able to link two transactions as being originated by the same user. For this definition, a transaction consists of the submission of a data report, the transmission to the queriers as well as the posting and collection of the incentives. Please note that this property is independent of the report data. For example, including a unique user identifier in every report payload would trivially enable report linking.

Both, PEPSICo and BBA+ include a similar privacy notion. However, a transaction in our extended scheme is more complex and it would be theoretically possible for the extended scheme to allow transaction linking, even if using instantiations of PEPSICo and BBA+ that do not. A trivial example would be to use the same bulletin nonce p for all data reports of the same user. We, therefore, have to provide a modified privacy definition for the extended scheme. We call the corresponding property *transaction unlinkability*.

This property implies that balance tokens are owner-hiding and balance-hiding. If the adversary could determine the owner of a balance token, he would trivially be able to link transactions. Furthermore, if the current balance could be determined from a token (during the accumulate protocol) a collusion of adversaries and the ISP could keep track of the possible token balances and use it as a side-channel to track users.

Confidentiallity of report data and query identifier

The service provider works as a mediator between the users and the queriers, mainly for efficiency purposes and to allow queriers to remain offline parties. Therefore, even though all the reports are routed through him, he should not be able to obtain information about their content, including the query identity which encodes the type of the report.

We require that the reported data and the query identity are only visible to authorized queriers. This property corresponds to the *node privacy* and *query privacy* properties of PEPSICo.

System security of the incentive mechanism

A secure incentive mechanism has to prevent cheating by the user as well as the system operator.

As this property is independent of the participatory sensing model it is used with, we can reuse the security notions from BBA+. We, therefore, require that balance tokens are *owner-binding* and *balance-binding* and that they guarantee freshness. The last property is split into two. *Double-spending detection* allows the ISP to determine whether the same token was used twice within a transaction and *false accusation protection* protects the user by requiring that a user can only be proved guilty of double-spending if he indeed committed this deed.

Security of the bulletin mechanism

The bulletin mechanism has to ensure that rewards posted on the bulletin board can only be claimed by the user that submitted the corresponding report, that is, knowledge of the bulletin secret is required to claim a reward from the bulletin board. To ensure this, we define the *false claim protection* property.

In summary, we require that the security notions of the underlying models hold and require that even the additional inter-dependencies in the extended model do not affect the users' privacy. In the following, we therefore only define *transaction unlinkability* and *false claim protection.*

6.4.2 Transaction unlinkability

To define transaction unlinkability, we modify the *privacy* property of BBA+, which follows the real/ideal world paradigm. The adversary plays the role of the SP and ISP. He has access to oracles that allow the creation and interaction with honest users as well as to corrupt users and queriers. In the real world, the oracles behave as the real users would. However, in the ideal world, the interaction is simulated without using user-related data (except for the issue process, which is identifying). The goal of the adversary is to decide whether he is in the real or the ideal world. If both worlds are indistinguishable for the adversary, he cannot learn any information in the real world that the simulator did not use in the ideal world as such information could be used to distinguish both worlds.

Definition 6.2 (Transaction unlinkability) *The real and ideal world experiments are defined in Figure 6.5 and the required oracles in Figure 6.3. An instantiation* PI *of our model is called* transaction unlinkable *if the real and the ideal world experiments are computational indistinguishable. More specific, for all* PPT $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$

$$\left|\Pr\left[\mathrm{Exp}^{\text{TU-real}}_{\mathrm{PI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\text{TU-ideal}}_{\mathrm{PI},\mathcal{A}}(n) \stackrel{?}{=} 1\right]\right| \leq \mathrm{negl}(n)$$

6.4.3 False claim protection

It should not be possible for an adversary (that does not collude with the ISP) to claim incentives that have been posted for a report where the adversary does not know the bulletin secret. More precise, no adversary can generate a valid bulletin secret from a bulletin nonce.

Within the definition of this property, the adversary takes the role of a user as the property protects the system and mainly other users from cheating users.

HonUser()

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathsf{UGen}(\mathsf{CRS})$
return $\mathsf{pk}_{\mathcal{U}}$

RealHonIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \stackrel{?}{=} \emptyset$ **then**
  $\mathfrak{I} \leftarrow \mathfrak{I} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  $\mathtt{Issue} \left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \right\rangle$
endif

RealHonReportData($\mathsf{pk}_{\mathcal{U}}, qid, m$)

if $\mathsf{regMN}_{qid,\mathcal{U}} \neq \bot$ **then**
  $\mathsf{regMN}_{qid,\mathcal{U}} \leftarrow \mathsf{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
  $\mathfrak{Q}_{\mathcal{U}} \leftarrow \mathfrak{Q}_{\mathcal{U}} \cup \{qid, \mathsf{regMN}_{qid,\mathcal{U}}\}$
endif
$(c, p, r) \leftarrow \mathsf{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid,\mathcal{U}}, qid, m)$
$\mathfrak{M}_{\mathcal{U}} \leftarrow \mathfrak{M}_{\mathcal{U}} \cup \{(qid, m, r)\}$
$\mathfrak{R}(p) := r$
return (c, p)

RealHonCollect($pk_{\mathcal{U}}, p, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ and $\mathfrak{R}(p) \neq \bot$ **then**
  $r = \mathfrak{R}(p)$
  $\mathtt{Collect} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{U}}, w_{\mathcal{U}}, r), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, p, v) \end{matrix} \right\rangle$
endif

RealHonRedeem($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ **then**
  $\mathtt{Verify} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{U}}, w_{\mathcal{U}}, v), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v) \end{matrix} \right\rangle$
endif

RealCorrupt($\mathsf{pk}_{\mathcal{U}}$)

return $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{U}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{M}_{\mathcal{U}})$

RegisterQ(qid)

return $\mathsf{regQ}_{qid} \leftarrow \mathsf{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

CorruptRA

return $\mathsf{sk}_{\mathcal{RA}}$

SimHonIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \stackrel{?}{=} \emptyset$ **then**
  $\mathfrak{I} \leftarrow \mathfrak{I} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  $\mathtt{Issue} \left\langle \mathsf{Sim}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \right\rangle$
endif

SimHonReportData($\mathsf{pk}_{\mathcal{U}}, qid, m$)

$\mathfrak{Q}'_{\mathcal{U}} \leftarrow \mathfrak{Q}'_{\mathcal{U}} \cup \{qid\}$
$\mathsf{regMN}_{qid} \leftarrow \mathsf{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$(c, p, r) \leftarrow \mathsf{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$
$\mathfrak{M}_{\mathcal{U}} \leftarrow \mathfrak{M}_{\mathcal{U}} \cup \{(qid, m, r)\}$
$\mathfrak{R}(p) := r$
return (c, p)

SimHonCollect($\mathsf{pk}_{\mathcal{U}}, p, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ and $\mathfrak{R}(p) \neq \bot$ **then**
  $r := \mathfrak{R}(qid, m, p)$
  $\mathtt{Collect} \left\langle \mathsf{Sim}(r), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, p, v) \right\rangle$
endif

SimHonRedeem($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathfrak{I} \cap \{\mathsf{pk}_{\mathcal{U}}\} \neq \emptyset$ **then**
  $\mathtt{Verify} \left\langle \mathsf{Sim}(w_{\mathcal{U}}, v), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v) \right\rangle$
endif

SimCorrupt($\mathsf{pk}_{\mathcal{U}}$)

$(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{U}}, \mathfrak{Q}_{\mathcal{U}}) \leftarrow \mathsf{Sim}(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \mathfrak{Q}'_{\mathcal{U}}, state_{\mathsf{Sim}})$
return $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{U}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{M}_{\mathcal{U}})$

Figure 6.3: Oracles for the *transaction unlinkability* experiments. For simplicity, the return values of the interactive protocol calls have been omitted. The Sim algorithm used by SimCorrupt is given in Figure 6.4. If a protocol run for a user identity is not accepted ($b_{\mathcal{U}} = 0$) then the user will reject to further participate, ie. $\mathcal{A}$ cannot make any more oracle calls to this user identity.

$\underline{\mathsf{Sim}(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \mathfrak{Q}'_{\mathcal{U}}, state_{\mathsf{Sim}})}$

$(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{U}}) \leftarrow \mathsf{Sim}_{\mathrm{BBA+}}(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, state_{\mathsf{Sim}})$
for $qid \in \mathfrak{Q}'_{\mathcal{U}}$ **do**
  $\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
  $\mathfrak{Q}_{\mathcal{U}} \leftarrow \mathfrak{Q}_{\mathcal{U}} \cup \{qid, \mathsf{regMN}_{qid}\}$
endfor
return $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{U}}, \mathfrak{Q}_{\mathcal{U}})$

Figure 6.4: Simulation algorithm for the SimCorrupt oracle

$\underline{\mathrm{Exp}^{\text{TU-real}}_{\mathrm{PI},\mathcal{A}}(n)}$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, s_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\text{HonUser,RegisterQ,CorruptRA,RealHonIssue,RealHonReportData,RealHonCollect,RealHonRedeem,RealCorrupt}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{RA}}, s_0)$
return b

$\underline{\mathrm{Exp}^{\text{TU-ideal}}_{\mathrm{PI},\mathcal{A}}(n)}$

$(\mathsf{CRS}, \mathsf{td}_{sim}) \leftarrow \mathtt{SimSetup}(1^n)$
$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, s_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\text{HonUser,RegisterQ,CorruptRA,SimHonIssue,SimHonReportData,SimHonCollect,SimHonRedeem,SimCorrupt}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{RA}}, s_0)$
return b

Figure 6.5: Real and ideal world experiments *transaction unlinkability*

$\underline{\mathrm{Game}^{\mathrm{FCP}}_{\mathrm{PI},\mathcal{A}}}$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, state_0) \leftarrow \mathcal{A}_0(\mathsf{pk}_{\mathcal{I}})$
$((\tau, b_{\mathcal{U}}, state_1), b_{\mathcal{I}}) \leftarrow \mathtt{Issue}\left\langle \mathcal{A}_1(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \right\rangle$
$\mathsf{regMN}_{qid} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$(c, p, r) \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$
$r' \leftarrow \mathcal{A}_2(state_1, c, p)$
$((\tau', b_{\mathcal{U}}, w'), (\mathsf{dstag}, hid, b_{\mathcal{I}})) \leftarrow \mathtt{Collect}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, r'), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, p, v) \right\rangle$
return $b_{\mathcal{U}} \neq 0$ or $b_{\mathcal{I}} \neq 0$

Figure 6.6: False claim protection game for the interim model

Definition 6.3 (False claim protection) *An instantiation* PI *of our model has* false claim protection, *if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, *for all* (qid, m) *and the game* $\text{Game}^{\text{FCP}}_{\text{PI},\mathcal{A}}$ *as defined in Figure 6.6*

$$\Pr\left[\text{Game}^{\text{FCP}}_{\text{PI},\mathcal{A}}(n) = 1\right] \leq \text{negl}(n)$$

6.5 Instantiation

In this section, we give a generic instantiation of the interim model, based on arbitrary instantiations of PEPSICo and BBA+. Subsequently, we prove that the security properties defined in Section 6.4 hold for the implementation.

6.5.1 Generic instantiation from PEPSICo and BBA+

We instantiate the interim model based on PEPSICo and BBA+. A collision-resistant hash function is used to implement the bulletin mechanism that allows the collection of incentives for a data report.

Definition 6.4 (Instantiation of interim I3PS) *Let* $H : \{0,1\}^* \to \{0,1\}^n$ *be a collision resistant hash function. We assume that the key* hk *is fixed by the implementation of the hash function and, therefore, write* H *instead of* H_{hk}. *Alternatively,* hk *could be generated as part of* $\mathtt{Setup}$ *and included in the* CRS. *Let further* $\text{BBA+} = (\mathtt{Setup}_{\text{BBA+}}, \mathtt{IGen}_{\text{BBA+}}, \mathtt{UGen}_{\text{BBA+}}, \mathtt{Issue}_{\text{BBA+}}, \mathtt{Accum}_{\text{BBA+}}, \mathtt{Verify}_{\text{BBA+}}, \mathtt{UVer}_{\text{BBA+}}, \mathtt{IdentDS}_{\text{BBA+}}, \mathtt{VerifyGuilt}_{\text{BBA+}})$ *be an BBA+ instantiation and* $\text{PI} = (\mathtt{Setup}_{\text{PI}}, \mathtt{RegisterMN}_{\text{PI}}, \mathtt{RegisterQ}_{\text{PI}}, \mathtt{ReportData}_{\text{PI}}, \mathtt{SubscribeQuery}_{\text{PI}}, \mathtt{ExecuteQuery}_{\text{PI}}, \mathtt{DecodeData}_{\text{PI}})$ *be an PEPSICo instantiation.*

Then the generic instantiation GI *of the interim model is given as follows:* $\mathtt{ReportData}$, $\mathtt{Collect}$ *and* $\mathtt{ExecuteQuery}$ *are defined in Figure 6.7 and the remaining algorithms and protocols are equivalent to their counterparts in PEPSICo and BBA+ as shown below.*

$\mathtt{Setup} = \mathtt{Setup}_{\text{BBA+}}$ $\quad$ $\mathtt{UVer} = \mathtt{UVer}_{\text{BBA+}}$

$\mathtt{SetupRA} = \mathtt{Setup}_{\text{PI}}$ $\quad$ $\mathtt{RegisterQ} = \mathtt{RegisterQ}_{\text{PI}}$

$\mathtt{IGen} = \mathtt{IGen}_{\text{BBA+}}$ $\quad$ $\mathtt{SubscribeQuery} = \mathtt{SubscribeQuery}_{\text{PI}}$

$\mathtt{UGen} = \mathtt{UGen}_{\text{BBA+}}$ $\quad$ $\mathtt{DecodeData} = \mathtt{DecodeData}_{\text{PI}}$

$\mathtt{Issue} = \mathtt{Issue}_{\text{BBA+}}$ $\quad$ $\mathtt{IdentDS} = \mathtt{IdentDS}_{\text{BBA+}}$

$\mathtt{RegisterMN} = \mathtt{RegisterMN}_{\text{PI}}$ $\quad$ $\mathtt{VerifyGuilt} = \mathtt{VerifyGuilt}_{\text{BBA+}}$

$\mathtt{Redeem} = \mathtt{Verify}_{\text{BBA+}}$

6.5.2 Security of the instantiation

In this section, we prove that the given implementation satisfies the specified security properties. As with the definitions, we only provide the proves for the adapted security properties defined in Sections 6.4.2 and 6.4.3. The other properties follow directly from the security of the underlying PEPSICo and BBA+ instantiations.

$\mathsf{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$

$r \xleftarrow{\$} \{0,1\}^n$
$p \leftarrow H(r)$
$c \leftarrow \mathsf{ReportData}_{\mathrm{PI}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$
return (c, p, r)

$\mathsf{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, , p, s)$

if $\mathsf{ExecuteQuery}_{\mathrm{PI}}(\mathsf{pk}_{\mathcal{RA}}, c, s) \neq \perp$ **then**
  return (c, p)
else
  return $\perp$

$\mathsf{Collect}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau, w, r), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, p, v)\right\rangle$

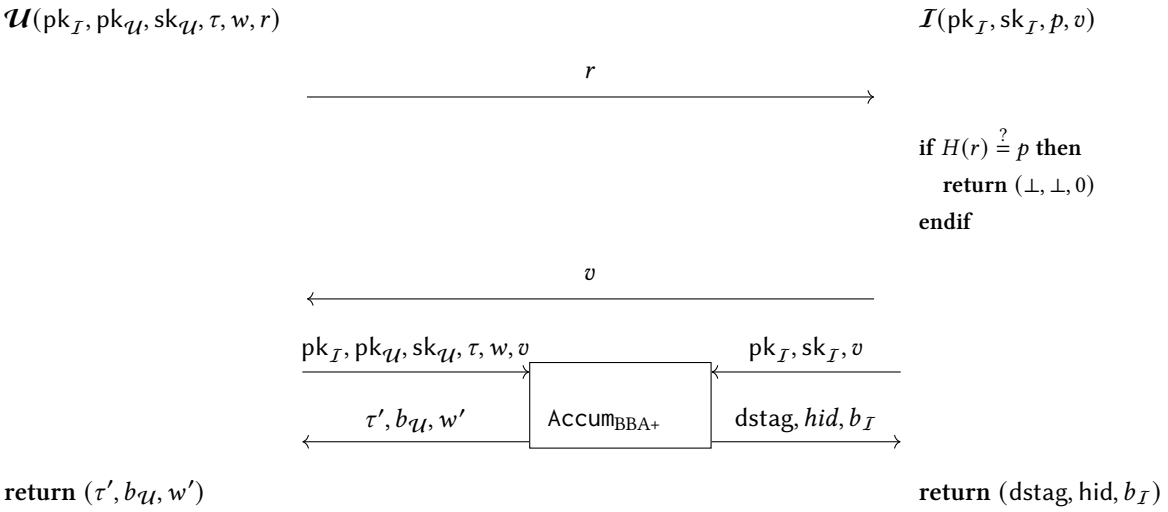

Figure 6.7: Implementations of ReportData, Collect and ExecuteQuery

Multiple report unlinkability

Before we can prove that our instantiation satisfies *report unlinkability*, we define *multiple report unlinkability* as a helper property. This property follows directly from the *report unlinkability* property of PEPSICo and therefore holds for our instantiation as proven below.

Definition 6.5 (Multiple report unlinkability) *An instantiation of the interim model I has* multiple report unlinkability *if an adversary cannot decide whether a list of data reports have been generated from a fixed registration value or from a fresh registration value for each data report. More precise, for the experiments defined in Figure 6.8 we require that for all* PPT *adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$*

$$\mathsf{Adv}^{\mathrm{mRU}}_{I,\mathcal{A}}(n) = \left|\Pr\left[\mathrm{Exp}^{\text{mRU-real}}_{I,\mathcal{A}}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\text{mRU-ideal}}_{I,\mathcal{A}}(n) \stackrel{?}{=} 1\right]\right| \leq \mathsf{negl}(n)$$

where q is the number of plaintexts the adversary requests data reports for.

$\mathrm{Exp}^{\text{mRU-real}}_{I,\mathcal{A}}(n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(qid, \{m_i\}_{0 \leq i < q}, state_0) \leftarrow \mathcal{A}_0(\mathsf{pk}_{\mathcal{RA}})$
$\mathsf{regMN} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
for $i := 0$ to $(q-1)$ **do**
  $c_i \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}, qid)$
endfor
$b \leftarrow \mathcal{A}_1(\{c_i\}_{0 \leq i < q}, state_0)$

$\mathrm{Exp}^{\text{mRU-ideal}}_{I,\mathcal{A}}(n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(qid, \{m_i\}_{0 \leq i < q}, state_0) \leftarrow \mathcal{A}_0(\mathsf{pk}_{\mathcal{RA}})$
for $i := 0$ to $(q-1)$ **do**
  $\mathsf{regMN}_i \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
  $c_i \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_i, qid)$
endfor
$b \leftarrow \mathcal{A}_1(\{c_i\}_{0 \leq i < q}, state_0)$

Figure 6.8: *Multiple report unlinkability* experiments in the real and ideal world

Lemma 6.6 (Multiple report unlinkability) *The generic implementation of the interim model as defined in Section 6.5.1 has* multiple report unlinkability. *For all* PPT *adversaries $\mathcal{A}$*

$$\mathsf{Adv}^{\mathrm{mRU}}_{\mathrm{GI},\mathcal{A}}(n) \leq \mathsf{negl}(n)$$

Proof We start from $\mathrm{Exp}^{\text{mRU-real}}_{\mathrm{GI},\mathcal{A}}$ and modify the experiment step by step, replacing the registration value used to generate the data report with a new registration value (see Figure 6.9). Therefore, in the qth step we reach $\mathrm{Exp}^{\text{mRU-ideal}}_{\mathrm{GI},\mathcal{A}}$. We show that Step i is indistinguishable from Step $i+1$ for the adversary by reducing a successful distinguisher between those two steps to a successful adversary on the *report unlinkability* property of PEPSICo.

Now, let us assume $\mathcal{A}$ can efficiently distinguish between Step j and Step j+1. We construct an adversary $\mathcal{A}'$ on *report unlinkability* as shown in Figure 6.10. $\mathcal{A}'$ is successful whenever $\mathcal{A}$ is. As the underlying PEPSICo scheme has *report unlinkability*, we have a contradiction and, therefore, no efficient $\mathcal{A}$ can distinguish between two subsequent steps.

$\underline{\mathrm{Exp}^{\mathrm{mRU}}_{\mathrm{Step\ j},\mathcal{A}}(n)}$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(qid, \{m_i\}_{0 \le i < q}, state_0) \leftarrow \mathcal{A}_0(\mathsf{pk}_{\mathcal{RA}})$
$\mathsf{regMN} \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
for $i := 0$ to $(j-1)$ **do**
  $\mathsf{regMN}_i \leftarrow \mathtt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
  $c_i \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_i, qid)$
endfor
for $i := j$ to $(q-1)$ **do**
  $c_i \leftarrow \mathtt{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}, qid)$
endfor
$b \leftarrow \mathcal{A}_1(\{c_i\}_{0 \le i < q}, state_0)$

Figure 6.9: Step-wise transition between the mRU experiments of the real and ideal world

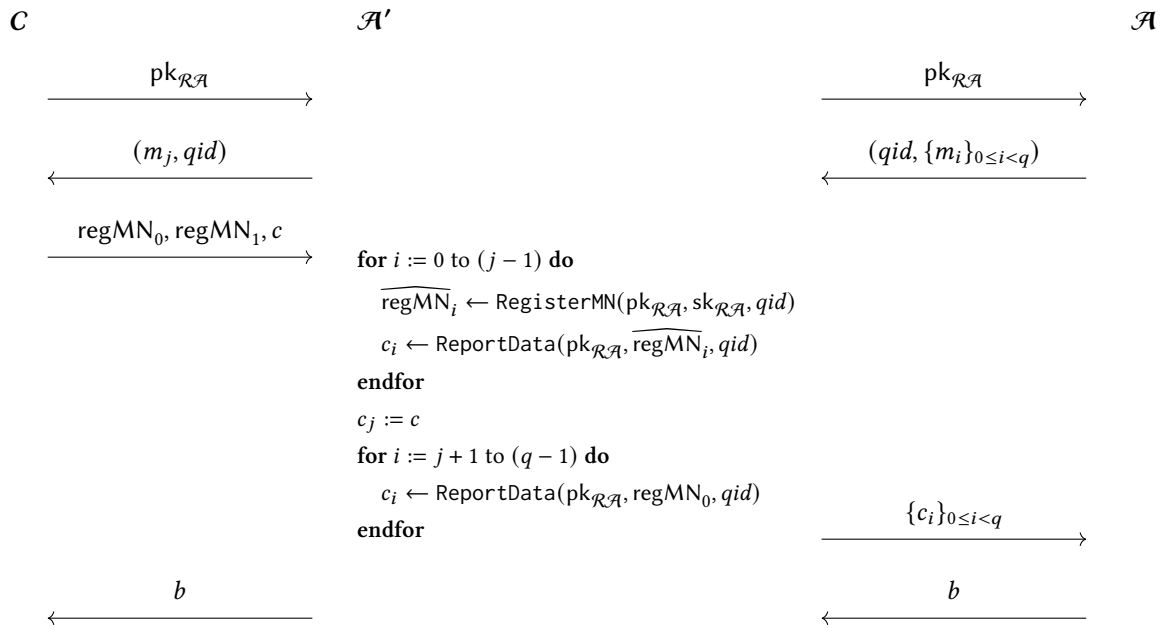

Figure 6.10: Reduction of an mRU adversary to an RU adversary

An adversary distinguishing between $\mathrm{Exp}^{\text{mRU-ideal}}_{\mathrm{GI},\mathcal{A}}$ and $\mathrm{Exp}^{\text{mRU-real}}_{\mathrm{GI},\mathcal{A}}$ has at most q times the advantage of an adversary distinguishing between two consecutive steps. For a generic instantiation GI of the interim model and PI being the underlying PEPSICo instantiation and for all PPT adversaries $\mathcal{A}$

$$\mathrm{Adv}^{\mathrm{mRU}}_{\mathrm{GI},\mathcal{A}}(n) \leq q \cdot \mathrm{Adv}^{\mathrm{RU}}_{\mathrm{PI},\mathcal{A}}(n)$$

As q is polynomial and $\mathrm{Adv}^{\mathrm{RU}}_{\mathrm{PI},\mathcal{A}}(n)$ is negligible, this advantage remains negligible.

Transaction unlinkability

Theorem 6.7 (Transaction unlinkability) *The generic instantiation* GI *as defined in Section 6.5.1 has* transaction unlinkability.

Proof The definition of *transaction unlinkability* is a modified version of the privacy proof of BBA+. We show that the modifications made to the real and ideal world experiments cannot be distinguished and, therefore, if the two experiments can be distinguished, the underlying BBA+ instantiation did not satisfy the *user privacy* notion. We show this by modifying the real and ideal world experiments of the original proof to match our definition of transaction unlinkability and proof that these modifications preserve the indistinguishability of the experiments.

We start from the BBA+ experiments $\mathrm{Exp}^{\text{PRIV-real}}_{\text{BBA+},\mathcal{A}}$ and $\mathrm{Exp}^{\text{PRIV-ideal}}_{\text{BBA+},\mathcal{A}}$. If BBA+ is *privacy-preserving*, those two experiments are indistinguishable.

First, we apply the changes that are the same in both worlds. We modify the two experiments to an interim version by generating the RA key pair as part of the setup and giving the adversary access to the RegisterQ and CorruptRA oracles from the *transaction unlinkability* experiment (Figure 6.3). This results in the experiments shown in Figure 6.11. We now assume there is an efficient adversary $\mathcal{A}$ that can distinguish between $\mathrm{Exp}^{\text{real}}_{\text{Step1},\mathcal{A}}$ and $\mathrm{Exp}^{\text{ideal}}_{\text{Step1},\mathcal{A}}$. We reduce this adversary to a successful adversary $\mathcal{A}'$ against the original BBA+ experiments. More precisely, we show that for all negligible functions negl

$$\begin{aligned} &\exists\ \mathrm{PPT}\ \mathcal{A} : \left|\Pr\left[\mathrm{Exp}^{\text{real}}_{\text{Step1},\mathcal{A}}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\text{ideal}}_{\text{Step1},\mathcal{A}}(n) \stackrel{?}{=} 1\right]\right| > \mathrm{negl}(n) \\ \Longrightarrow\ &\exists\ \mathrm{PPT}\ \mathcal{A} : \left|\Pr\left[\mathrm{Exp}^{\text{PRIV-real}}_{\text{BBA+},\mathcal{A}'}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\text{PRIV-ideal}}_{\text{BBA+},\mathcal{A}'}(n) \stackrel{?}{=} 1\right]\right| > \mathrm{negl}(n) \end{aligned}$$

As the existence of such an adversary contradicts with the *transaction unlinkability* property of BBA+, we have shown that our assumption was wrong and, therefore, the experiments are indistinguishable.

The reduction is given in Figure 6.12. During the setup phase, where there is no oracle access, $\mathcal{A}'$ generates the RA key pair and sends the RA's public key to the adversary. After the setup phase has been completed, $\mathcal{A}'$ has to simulate the RegisterQ and CorruptRA oracles for $\mathcal{A}$. As $\mathcal{A}'$ learns the RA's secret key, this can be done directly by executing the corresponding oracle code. All other oracles

$\mathrm{Exp}^{\mathrm{real}}_{\mathrm{Step1},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, s_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\mathrm{HonUser,RegisterQ,CorruptRA,RealHonIssue,RealHonAcc,RealHonVer,RealCorrupt}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{RA}}, s_0)$
return b

$\mathrm{Exp}^{\mathrm{ideal}}_{\mathrm{Step1},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}_{sim}) \leftarrow \mathtt{SimSetup}(1^n)$
$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathtt{SetupRA}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, s_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\mathrm{HonUser,RegisterQ,CorruptRA,SimHonIssue,SimHonAcc,SimHonVer,SimCorrupt}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{RA}}, s_0)$
return b

Figure 6.11: Interim experiments for the *transaction unlinkability* proof

available to $\mathcal{A}$ are already available to $\mathcal{A}'$ and therefore queries to these oracles just need to be relayed. Therefore, $\mathcal{A}'$ simulates the changes perfectly and has the same advantage as $\mathcal{A}$.

Second, we modify our interim experiments to equal our definition of *transaction unlinkability*. While the setup phase remains the same, we modify the oracles that are accessible by $\mathcal{A}$ afterward. $\mathcal{A}$ gains additional access to an oracle encapsulating the report data algorithm (RealHonReportData in the real world, SimHonReportData in the ideal world). Moreover, the oracles for the `Accum` and `Verify` protocols from BBA+ are replaced with oracles encapsulating the `Collect` and `Redeem` protocols of the interim model. More specific, in the real world we replace RealHonAcc with RealHonCollect and rename RealHonVer to RealHonRedeem. In the ideal world, we replace SimHonAcc with SimHonCollect and rename SimHonVer to SimHonRedeem. Additionally, the user corruption oracles RealCorrupt and SimCorrupt have to be modified.

Again, we assume the existence of an efficient adversary $\mathcal{A}$ successfully distinguishing between our two experiments, $\mathrm{Exp}^{\text{TU-real}}_{\mathrm{PI},\mathcal{A}}$ and $\mathrm{Exp}^{\text{TU-real}}_{\mathrm{PI},\mathcal{A}}$. We now show by reduction that this would imply the existence of a successful adversary on either the interim experiments or the *multiple report unlinkability* of the interim model (cf. Lemma 6.6). More precisely, we show that for all negligible functions negl

$$\begin{aligned}
&\exists\ \mathsf{PPT}\ \mathcal{A} : \left|\Pr\left[\mathrm{Exp}^{\text{TU-real}}_{\mathrm{PI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\text{TU-real}}_{\mathrm{PI},\mathcal{A}}(n) \stackrel{?}{=} 1\right]\right| > \mathsf{negl}(n) \\
\Longrightarrow\ &\exists\ \mathsf{PPT}\ \mathcal{A}' : \left|\Pr\left[\mathrm{Exp}^{\mathrm{real}}_{\mathrm{Step1},\mathcal{A}'}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{ideal}}_{\mathrm{Step1},\mathcal{A}'}(n) \stackrel{?}{=} 1\right]\right| > \mathsf{negl}(n) \\
&\vee\ \exists\ \mathsf{PPT}\mathcal{A}'' : \mathrm{Adv}^{\mathrm{mRU}}_{\mathrm{GI},\mathcal{B}}(n) > \mathsf{negl}(n)
\end{aligned}$$

Our new reduction uses the interface available to the adversary in the previous step. Basically,

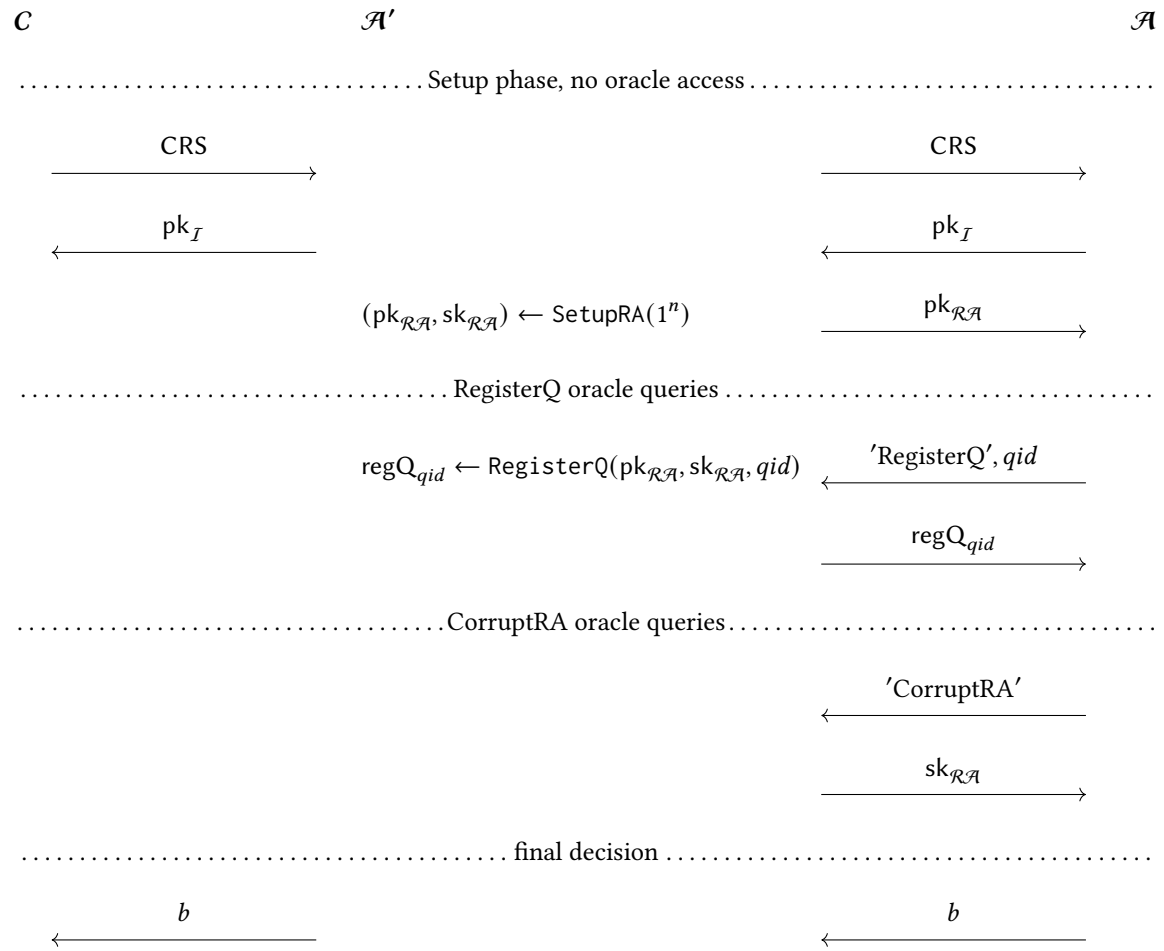

Figure 6.12: Reduction proof for the interim step: We reduce an adversary on the indistinguishability between $\mathrm{Exp}^{\mathrm{real}}_{\mathrm{Step1},\mathcal{A}}$ and $\mathrm{Exp}^{\mathrm{ideal}}_{\mathrm{Step1},\mathcal{A}}$ to and adversary on the indistinguishability between $\mathrm{Exp}^{\text{PRIV-real}}_{\text{BBA+},\mathcal{A}'}$ and $\mathrm{Exp}^{\text{PRIV-ideal}}_{\text{BBA+},\mathcal{A}'}$. If $\mathcal{A}$ queries an oracle other than RegisterQ or CorruptRA, the query is relayed to the corresponding oracle available to $\mathcal{A}'$.

SetupRA will now be executed by the challenger and the reduction receives $\mathsf{pk}_{\mathcal{RA}}$. We therefore only need to simulate the modified or added oracle queries.

The simulation for the RealHonReportData/SimHonReportData oracles is given in Figure 6.13. For the simulation, the RA's secret key is required which can be obtained by calling the CorruptRA oracle. In comparison to the previous oracle simulations, we now have the difficulty that there are different oracles for the real and ideal world. However, our reduction $\mathcal{A}'$ does not know in which world the game is played. We, therefore, use the behavior from the ideal world in our simulation and show that RealHonReportData and SimHonReportData are indistinguishable. The difference between those oracles is that the real world oracle only once requests a registration value for a fixed *qid* and user, while the simulation oracle requests a new registration value each time, independent of the user. Both oracle queries output a data report which has been generated by the underlying PEPSICo's ReportData algorithm. As our PPT adversary can use the oracle a polynomial number of times, this difference corresponds to the previously defined *multiple report unlinkability* property and by Lemma 6.6, $\mathcal{A}$ can distinguish between these two ways to generate reports with at most negligible advantage. The second output, the bulletin nonce p, is in both cases generated by applying the same hash function to a randomly chosen secret r. Therefore, p is equally distributed in both worlds and an adversary cannot distinguish in which world it has been created. $\mathfrak{Q}'_{\mathcal{U}}$, $\mathfrak{M}_{\mathcal{U}}$ and $\mathfrak{R}$ are used to store information that is required to simulate the other oracles.

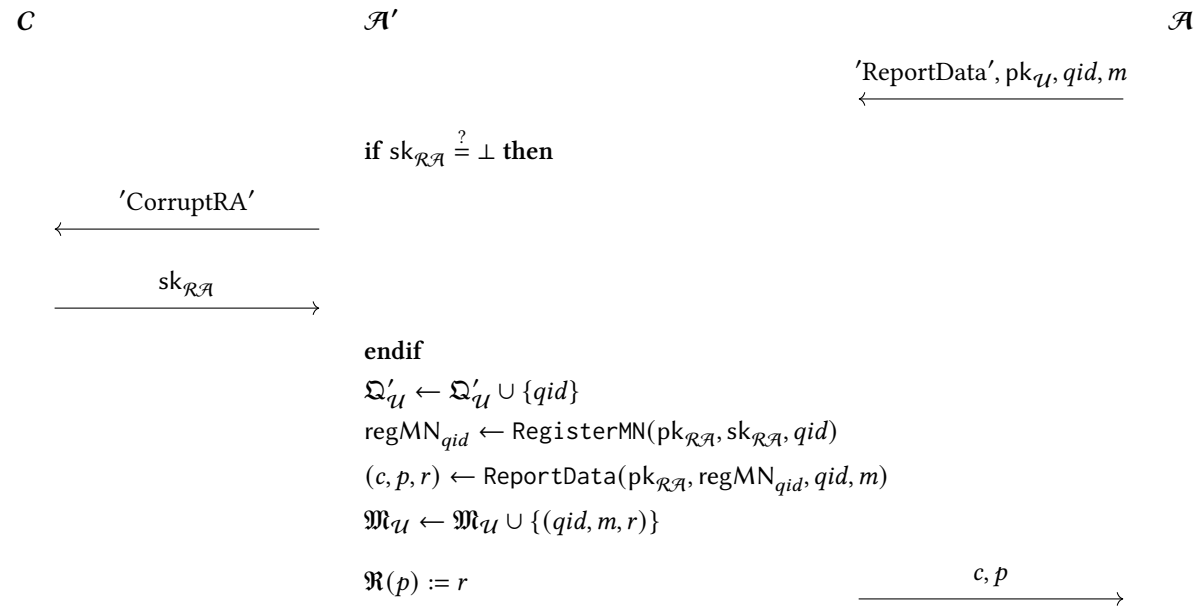

Figure 6.13: Simulation of RealHonReportData/SimHonReportData oracle queries

Figure 6.14 shows how the RealHonCollect/SimHonCollect oracle queries are simulated based on the RealHonAcc/RealHonVer oracles available to the reduction. In addition, $\mathcal{A}'$ has to keep track of the list $\mathfrak{I}$ of user public keys for which a token has been issued. This is done by remembering all $\mathsf{pk}_{\mathcal{U}}$s for

which $\mathcal{A}$ calls RealHonIssue/SimHonIssue. The simulation retrieves the bulletin secret r corresponding to the p specified by the adversary. Here, the applied modifications are the same in the real and ideal world. Moreover, $\mathfrak{R}$ is identically defined in both worlds. Therefore, our simulation is perfect.

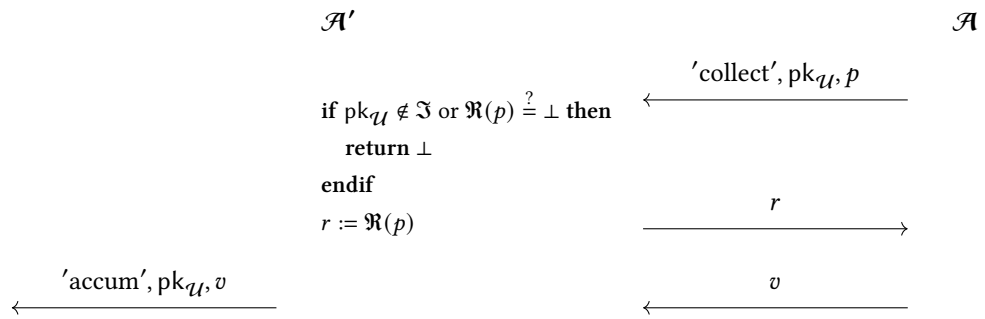

Figure 6.14: Simulation of RealHonCollect/SimHonCollect oracle queries

The simulation for the modified RealCorrupt/SimCorrupt oracles is given in Figure 6.15. Again, we use the behavior from the ideal world within the simulation and show that it is indistinguishable from the real world, at least for the modified parts. As $\mathfrak{M}_{\mathcal{U}}$ is identically defined in both worlds, the difference lies within $\mathfrak{Q}_{\mathcal{U}}$. As regMN_{qid} is independent of the user's identity, $\mathfrak{Q}_{\mathcal{U}}$ is identically distributed in both worlds. However, in the ideal world, the data reports that have been generated by SimHonReportData beforehand have been generated using different registration values, while in the real world, the exposed registration value has been used to generate all the data reports for the corresponding *qid*. However, this difference does not allow an adversary to distinguish both worlds because the *report unlinkability* property of the underlying PEPSICo scheme (Section 4.2.3) guarantees that it is not possible to distinguish which registration value has been used to create a report (cf. Lemma 6.6).

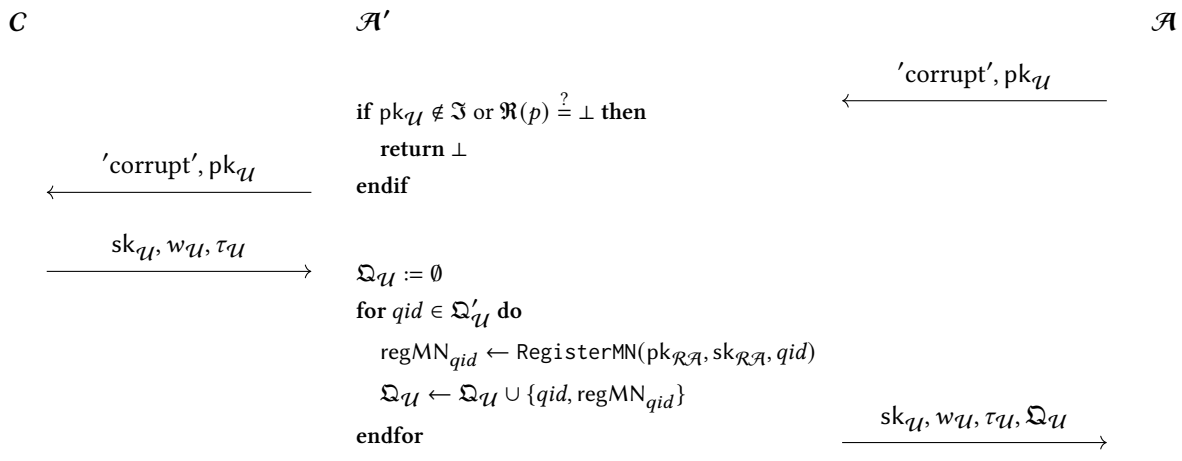

Figure 6.15: Simulation of RealCorrupt/SimCorrupt oracle queries

Finally, the reduction has to relay the decision of the adversary $\mathcal{A}$, whether he is living in the real or the ideal world, to the challenger. Now, as long as $\mathcal{A}'$ cannot break the *multiple report unlinkability* property of PI, our reduction can distinguish between $\text{Exp}^{\text{real}}_{\text{Step1},\mathcal{A}}$ and $\text{Exp}^{\text{ideal}}_{\text{Step1},\mathcal{A}}$, and therefore between $\text{Exp}^{\text{PRIV-real}}_{\text{BBA+},\mathcal{A}}$ and $\text{Exp}^{\text{PRIV-ideal}}_{\text{BBA+},\mathcal{A}}$, with the same advantage as $\mathcal{A}$, As, by assumption, this advantage is non-negligible, this contradicts with the *privacy* property of BBA+.

False claim protection

To show *false claim protection* for the instantiation defined in Section 6.5.1, we can show that a successful attacker would violate the *preimage resistance* of the underlying hash function H.

Theorem 6.8 (False claim protection) *The generic instantiation* GI *as defined in Section 6.5.1 has* false claim protection.

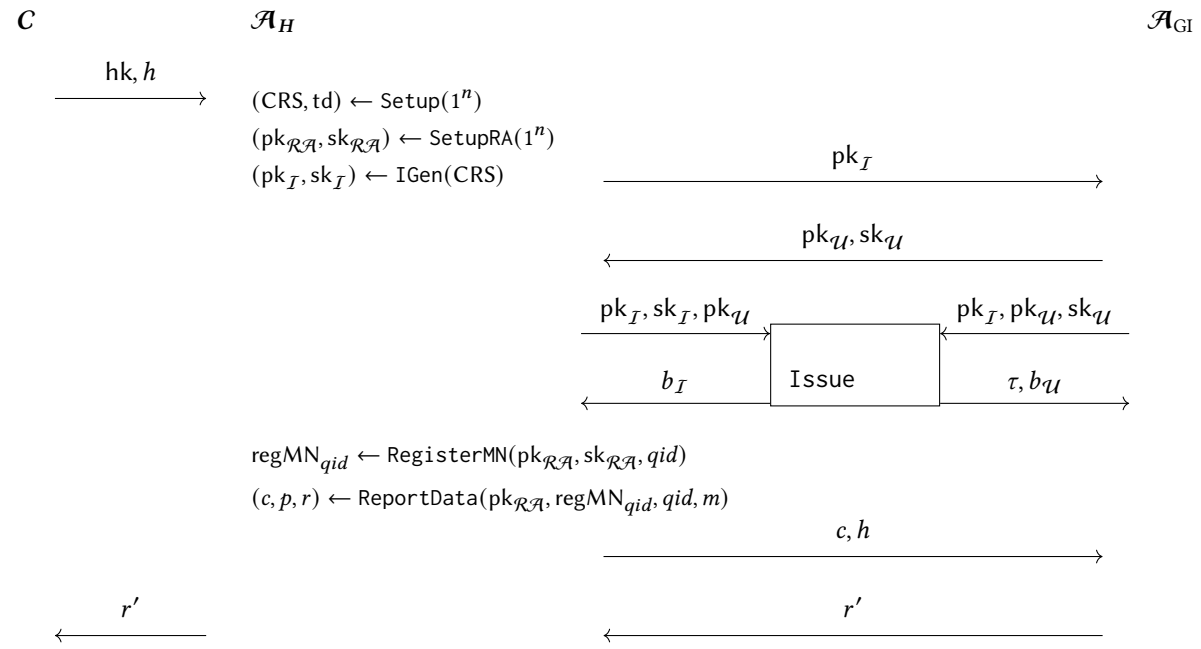

Figure 6.16: Reduction of a successful adversary on the *false claim protection* of the generic implementation to an adversary on the *preimage resistance* of the used hash function

Proof We construct an adversary $\mathcal{A}_H^{\mathcal{A}_{\text{GI}}}$ on the hash function H as follows: $\mathcal{A}_H$ simulates the *false claim protection* game for $\mathcal{A}_{\text{GI}}$ and replaces the bulletin nonce p with the challenge h from the *preimage resistance* game (Figure 6.16). As r was chosen independently at random, this is indistinguishable for $\mathcal{A}_{\text{GI}}$ from the real game. Now, if $\mathcal{A}_{\text{GI}}$ manages to win in the *false claim protection* game, it has to hold that $H(r') = h$ where r' is the bulletin secret $\mathcal{A}_{\text{GI}}$ outputs. Therefore, $\mathcal{A}_H$ wins the *preimage resistence*

game with the same probability as $\mathcal{A}_{\mathrm{GI}}$, which has to be negligible for a *preimage resistant* H. More formally, for all adversaries $\mathcal{A}_H, \mathcal{A}_{\mathrm{GI}}$

$$\Pr\left[\mathrm{Game}^{\mathrm{PR}}_{H,\mathcal{A}_H}(n) = 1\right] \leq \mathrm{negl}(n) \implies \Pr\left[\mathrm{Game}^{\mathrm{FCP}}_{\mathrm{GI},\mathcal{A}_{\mathrm{GI}}}(n) = 1\right] \leq \mathrm{negl}(n)$$

6.6 Discussion

In this chapter, we defined a participatory sensing model as a straight-forward combination of PEPSICo and BBA+. Our model extends the infrastructure of PEPSICo by the possibility to reward users with incentives to make contributing more attractive. Moreover, it preserves strong privacy properties to protect users from being tracked by the platform and queriers.

However, looking at the afore-defined model, we identified the following shortcomings:

Double reporting

One essential problem of the model is the missing protection from double reporting. There is nothing to prevent a user from transmitting the same data report twice. While it might be possible for queriers to detect such behavior for very specific query types, this is not the general case. There is no way for a querier to decide whether a tuple of reports with identical data originated from the same user or two different users. On one hand, this is a wanted privacy feature but on the other hand, it allows a user to send a report multiple times to earn more incentives. This problem did not exist within the original PEPSICo scheme as users had no motivation for such behavior. The only motivation users had at all for participating was to support the querier with his research.

Handling of bad data reports

There is an additional problem with bad data reports. The general technique to prevent bad data reports is a reputation mechanism, which is out of our scope for now. However, in our model, all data reports are saved by the SP and have to be checked for matches with every new subscription token. A feedback mechanism would allow queriers that have determined a report to have bad data quality to report this to the SP which could delete those reports if multiple queriers agree or if the query identity is exclusively used by this querier.

A supporting factor for this issue is that the message space of a data report does not depend on the query identifier. It is therefore not possible to restrict the message space of a data report to possible sensing results.

Timeliness of rewards

In our model, after a user submits a report, an unspecified amount of time might pass before a querier comes online to collect the reports for the query identity in question. Even more, time may pass until the querier analyzed the data quality and instructed the ISP to reward the user with incentives. This behavior might be an inhibitor as people in today's world are used to get

instant feedback on their actions. For the queriers, on the other hand, it might be more convenient to collect and analyze the cumulated data reports once a month, resulting in long waiting times for users.

Incentive sharing

BBA+ prevents users from uncontrolled points transfer. Even though the main reason for this is to support use cases where negative points can be collected, all transactions in the model should be handled by the ISP. However, as the bulletin mechanism that is used to distribute incentive rewards does not depend on a long term secret, sharing the bulletin secret would be comparable to transferring a specific number of incentives between users. Furthermore, this issue is critical towards the double reporting problem. As long as incentive sharing is possible, a malicious user could bypass any report limit by creating a new user identity, registering it for the query identity, submitting the report and claiming the corresponding incentives for his original user identity. This assumes that there are no restrictions for the registration of a new user in place. However, without the possibility of incentive sharing, this approach would lead to many balance tokens with a low balance, which would potentially be of limited value.

Verifiable delivery

In the interim model, there is no measure to ensure that users receive honest and fair incentive rewards for their data reports. However, such a property is difficult to enforce. As we discussed in Section 6.1.1, only the queriers can determine the quality of a data report and therefore decide on the appropriate number of incentives as compensation. However, this makes it impossible to prevent the querier from cheating. We argue that it is within the querier's interest to distribute fair rewards as their single purpose is to attract users in providing more data. A possible countermeasure outside the model could be to provide the infrastructure for a public rating of queriers by the users.

However, it remains for the querier to trust the ISP to correctly deliver posted incentives. To weaken this trust assumption, an additional mechanism could be introduced that allows queriers to verify the correct delivery of incentives.

7 Advanced model

While the interim model achieves the goal of combining PEPSICo and BBA+ to a participatory model with an incentive mechanism, it falls short in addressing the issues that arise with the combination (cf. Section 6.6). We, therefore, specify an advanced model addressing the identified issues as feasible.

We first discuss how we address the identified issues. We then provide an overview of our model (Section 7.2), followed by a formal definition (Section 7.3). In Section 7.4, we define the security properties for our model, before we give an instantiation in Section 7.5 and proof that it indeed meets the specified security properties in Section 7.6. Subsequently, in Section 7.7, we argue that our instantiation is sufficiently efficient to be used within real-world applications. Lastly, in Section 7.8 we provide a discussion of our model.

7.1 Improvements over the interim model

7.1.1 Prevention of incentive sharing

The first issue that should be addressed is the problem of incentive sharing as it correlates with the double-reporting issue. However, if we restrain from using BBA+ as a black box, we can replace the bulletin nonce with a commitment to the secret key of the user. We then modify the accumulation protocol to include the verification that the commitment is indeed a commitment to the same secret key that has been used to create the balance token. Therefore, the user identity under which the incentives posted for a report can be collected is fixed at the time of report submission. Because of the hiding properties of the commitment and the zero-knowledge property of the involved proof, transactions remain unlinkable.

7.1.2 Prevention of double-reporting

The complete prevention of double-reporting is nearly impossible to achieve, especially if we do not want to limit our system to a specific use case. With rising data complexity, it gets more and more unlikely that two honest reports contain identical data and this could be used as an indicator, but for queries that have only a small set of valid sensing data, such as yes/no-queries, this is not possible. Ultimately, it remains the responsibility of the querier to decide whether the reported data is trusted or not and what number of incentives it is worth. However, we want to keep the required effort at the querier's site, and with it the hurdle of becoming a querier, low, if possible.

One approach to restrain the extent of the double-reporting problem is to limit the number of reports a single user can generate for a fixed query identity. More precisely, we enable the specification of an upper limit of data reports allowed to be submitted for a specific query identity by the same user. Note that because of the correlation with the incentive sharing problem, we have to make sure that reports are counted under the same user identity as contained within the bulletin commitment.

There are multiple options to design such a mechanism. One of the core difficulties is, that the SP should not know the query identity of a data report and queries should remain unlinkable even if the RA colludes with the other parties. Multiple tradeoffs between security, flexibility, and efficiency are possible. In the following, we discuss two possible approaches.

One-time token mechanism

The idea behind this mechanism is, given a query identity and a per-user contribution limit for this identity, to provide a registering user with an equal number of one-time tokens. Each time a report is submitted, a one-time token has to be attached. Furthermore, SP checks the validity and freshness of the token. The required protocols together with a possible instantiation are given in Appendix A.

The disadvantage of this approach is that it is only feasible for small report limits per query identity. We argue that this is likely to be the case for most use cases where the number of reports per user should be limited. However, we require an additional mechanism to support query identities without a report limit.

The RA could maintain a whitelist of query identities that do not specify a limit. Limiting the space of query identities to $\mathbb{Z}_p$, which we require for the zero-knowledge proofs anyway, would allow using a cryptographic accumulator for membership testing on this whitelist. When submitting a report, the user would have either to supply a valid one-time-token or a zero-knowledge proof attesting that the query identity the report has been generated for is indeed a member of the whitelist.

Report counter with BBA+

A second possibility is to use a modified version of BBA+. Upon registering for a query identity, the RA issues a BBA+ token for this query identity to the user, incorporating the maximum number of data reports he is allowed to send. This number is used as a counter. It is decreased by one upon the submission of a report for this *qid*. Reports can only be submitted as long as the counter is larger than 0. This construction avoids expensive range proofs as a check for inequality is sufficient. Note that each token needs to be uniquely linked to a specific query identity.

This approach is much more flexible as the one-time token mechanism. Increasing the report maximum comes at no additional costs. Moreover, we do not require separate handling of query identities without a support limit. In such a case, we can set the counter to its maximum, which will be large enough assuming that users only submit a polynomial number of reports. Therefore, we use this approach within our model.

7.1.3 Improved handling of bad data reports

We include a feedback mechanism, allowing queriers to report bad data reports to the service provider. The service provider should have a policy whether they are deleted directly or only if a certain number of queriers reports them as bad. We do not include a reputation mechanism in our model but recommend it as future work.

7.1.4 Timeliness of rewards

We do not address this issue within our model. Not enforcing immediate responses of the queriers is an essential design decision of PEPSICo that we maintain within our model. However, it could be beneficial for queriers to be transparent about the time intervals in which they collect and evaluate the data reports.

7.1.5 Verifiable delivery

Apart from the double-reporting problem, the other larger issue our model has to solve is how to provide verifiable delivery. The service provider should be able to prove to the querier that the required amount of incentives has indeed been delivered to the right user.

This property is correlated with the *node privacy* property of PEPSICo, that is, it is inherently unsolvable if the adversary can collude with a querier registered for the query identity in question. If the adversary can obtain the report data m, he can generate a new report for m by a compromised user and replace the original report. Therefore, this property is of limited value if there are no controls in place to restrict querier registrations. However, it works fine for query identities which are restricted to a set of queriers trusting each other not to collude with the SP.

To achieve verifiable delivery, upon submitting a report, the user includes a newly generated MAC key in the plaintext for the PEPSICo report. Therefore, only authorized queriers can obtain this key. When collecting incentives, a MAC tag is computed under this key, attesting that the incentives have been delivered correctly. Because of the security of the MAC, the SP cannot forge a valid tag. Note that a new key has to be used with each submitted report to maintain transaction unlinkability. We use a MAC instead of a signature scheme, as signature schemes usually have expensive key generation algorithms. Moreover, even using a signature scheme, we could not just send the public key in the clear. It would have to be bound to the content of the PEPSICo report in some way, else the SP could just replace it with a different key.

7.1.6 Transaction unlinkability against a malicious registration authority

The *report unlinkability* property of PEPSICo only holds towards an honest-but-curious RA, as the mobile node registration values are computed by the experiment. In the generic instantiation of PEPSICo from IBE, these values are computed independently of the user's identity using a PRF. We replace the PRF with a VRF. Together with the mobile node registration value, the RA provides a proof to the user

that the value has been computed correctly. This enables the user to detect if the RA deviates from an honest behavior with respect to the computation of the mobile node registration value, thus, we can provide *transaction unlinkability* against a malicious RA. Note that, in addition, we also get forward privacy for PEPSICo reports. As all users registered for a query identity use the same mobile node registration value, the adversary cannot link data reports to specific users based on previous knowledge of their mobile node registration value.

7.2 Overview of the model

Incorporating the improvements discussed above, we construct a new participatory model based on PEPSICo and BBA+. Our new model has the same parties than the interim model, namely:

Trusted Third Party (TTP)

The TTP is required for the trusted setup of the CRS, describing the algebraic framework for the incentive and report limitation mechanisms (BBA+).

Registration Authority (RA)

The registration authority is a semi-trusted party (trusted not to collude with the service provider to circumvent the confidentiality of report data and query identities, but not trusted regarding transaction unlinkability). It manages query identities, allowing users and queriers to register for a specific query identity to submit and collect reports for this query identity, respectively.

Service Provider (SP)

The SP acts as a mediator between the users and the queriers, allowing them to be modeled as offline entities. They store all the reports submitted by the users, validating the reporting limits, and make them available to be collected by registered queriers. Therefore, the SP has to be an online entity, always available to interact with users or queriers.

Incentive System Provider (ISP)

The ISP manages the incentive mechanism, allowing the users to collect incentive rewards provided to them by queriers and to redeem their collected incentives. Therefore, the ISP has to be an online entity.

User

Users are the participants that provide the data for the sensing queries. Therefore, they register for one or more query identity and submit reports for this query identity to the SP. Users are offline entities, meaning that they do not have to be constantly connected to the network. Therefore, they can only take part in communication if they are the initiator.

Querier

Queriers are the entities interested in receiving sensor reports. Therefore they register for one or more query identities and regularly collect the corresponding reports from the SP. They are

responsible for determining the quality of submitted data reports and reward users with incentive points for those data reports. Similar to users, queriers are offline entities.

Initially, the TTP has to generate and publish the CRS of the system and the RA and ISP have to generate their public and secret key pairs. Moreover, The registration authority has to generate a second key pair for the limitation mechanism shared with the service provider.

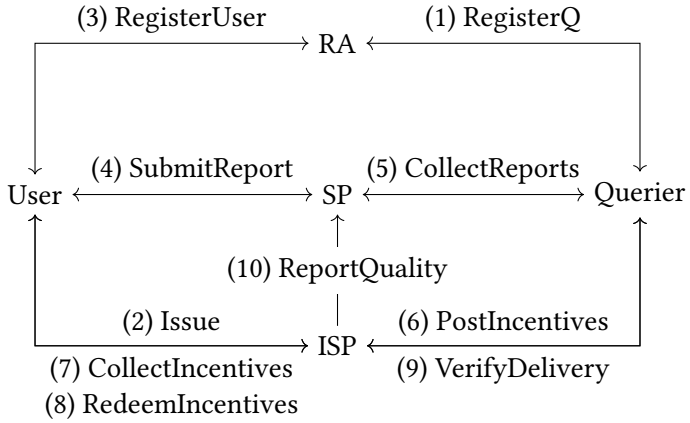

Figure 7.1: Overview of I3PS

Figure 7.1 outlines the system's architecture and the interactions between the parties. During the operation of the system, queriers can register themselves to receive reports for specific query identities with the RA (1). The query identity defines the collected data. To participate in the system, a user has to create his private and public key pair and needs to be issued an incentive token by the ISP (2). This token represents the users' incentive account, allowing them to accumulate and redeem incentive rewards. To submit reports of a specific type, the user has to register for the corresponding query identity once (3), obtaining a token representing the number of reports he is allowed to submit for this query identity. Reports can then be submitted to the SP (4), where they can be collected by queriers registered for the same query identity (5). Apart from using the reported data for their research, queriers can post incentive rewards for the users that submitted a report to the bulletin board maintained by the ISP (6). There, they can be collected by the corresponding users (7). They are accumulated in the users' incentive accounts represented by their incentive tokens. Furthermore, the ISP allows users to redeem incentive points they accumulated previously (8) to obtain rewards outside the system. The ISP can proof to the querier that the posted incentive rewards have been transferred to the corresponding user (9). Lastly, when posting incentives, the querier also indicates the quality of the data report. This allows the ISP to regularly give reports to the SP (10), allowing the deletion of bad data reports.

7.3 Formal definition

Definition 7.1 (I3PS) *An I3PS scheme consists of the following algorithms and protocols:*

Setup algorithms

$(\mathsf{CRS}, \mathsf{td}) \xleftarrow{\$} \mathtt{Setup}(1^n)$

The Setup algorithm for the incentive scheme is executed by the TTP once before the scheme can be used. The common reference string CRS *is made public while the trapdoor* td *remains a secret only used to define the security of the scheme.*

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \xleftarrow{\$} \mathtt{SetupRA}(\mathsf{CRS})$

With this algorithm, the RA creates its public and private key pair $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}})$*. The private key allows the RA to generate registration tokens for users and queriers. Additionally, it generates the key pair for the report limitation mechanism* $(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$ *which it shares with the SP. For convenience, it is assumed that* $\mathsf{pk}_{\mathcal{SP}}$ *contains* CRS*.*

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

This algorithm is executed by the ISP once to generate its public and private key pair. For convenience, it is assumed that $\mathsf{pk}_{\mathcal{I}}SP$ *contains* CRS*.*

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$

Every user is required to generate a public and private key. The public key $\mathsf{pk}_{\mathcal{U}}$ *acts as an identifier for the user.*

User protocols

$((\tau_{\mathcal{I}}, b_{\mathcal{U}}), b_{\mathcal{I}}) \leftarrow \mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \right\rangle$

The next step for a user would be to request a balance token. This is done by engaging in the issue protocol with the ISP. During this protocol, the identity of the user is exposed.

$$((\tau_{\mathcal{SP}}, b_{\mathcal{U}}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterUser}\left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \\ \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \end{matrix} \right\rangle$$

When registering for a query identity qid, the RA has to check the identity of the user and that the user is not already registered. Therefore, the protocol is identifying. I addition to the registration value regMN_{qid}*, the user obtains a report counter token* $\tau_{\mathcal{SP}}$*, incorporating* regMN_{qid}*, and* $ctr = max_{qid}$*, the maximum number of reports the user is allowed to submit.*

$$\begin{pmatrix} (\tau_{\mathcal{V}}, \tau^*_{\mathcal{SP}}, b_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{SP}}) \end{pmatrix} \leftarrow \mathtt{SubmitReport}\left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}, m), \\ \mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix} \right\rangle$$

This protocol is used to submit a report to the SP. The counter token $\tau_{\mathcal{SP}}$ *allows validating if the user is still within the limit of allowed reports for this qid. Apart from the report c, the SP outputs the public bulletin commitment* $com_{\mathcal{V}}$*.* dstag *and* hid *are required for double-spending detection. The user outputs the bulletin token* $\tau_{\mathcal{V}}$*, which is required to claim the incentives and contains the bulletin commitment* $com_{\mathcal{V}}$ *and the message authentication key* $\mathsf{k}_{\mathrm{mac}}$*, and a new counter token* $\tau^*_{\mathcal{SP}}$ *with the counter value from* $\tau_{\mathcal{SP}}$ *reduced by one.* $b_{\mathcal{U}}$ *and* $b_{\mathcal{SP}}$ *are used to indicate protocol failures.*

$((\tau_{\mathcal{I}}^*, b_{\mathcal{U}}, w^*), (t_{\text{mac}}, \text{dstag}, \text{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Collect} \left\langle \begin{matrix} \mathcal{U}(\text{pk}_{\mathcal{I}}, \text{pk}_{\mathcal{U}}, \text{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, \tau_{\mathcal{V}}, m), \\ \mathcal{I}(\text{pk}_{\mathcal{I}}, \text{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v) \end{matrix} \right\rangle$

With this algorithm, a user with a balance token τ with balance w and a bulletin secret $\tau_{\mathcal{V}}$ can claim any incentives posted for the corresponding nonce. After the protocol, the user has an updated token τ^* with the balance w^* that should be equal to $w + v$, whereby v is the incentive value that has been posted. Furthermore, the ISP obtains a MAC tag t_{mac} which allows to proof the correct delivery of the incentives to the querier that posted them and a double-spending tag dstag *which can be used to detect cheating users. The hidden user id* hid *is only required for definitional purpose.*

$((\tau'_{\mathcal{I}}, b_{\mathcal{U}}, w'), (\text{dstag}, \text{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Redeem} \left\langle \mathcal{U}(\text{pk}_{\mathcal{I}}, \text{pk}_{\mathcal{U}}, \text{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, v), \mathcal{I}(\text{pk}_{\mathcal{I}}, \text{sk}_{\mathcal{I}}, w, v) \right\rangle$

This algorithm allows a user to redeem a specific number of incentives from his token's balance. This algorithm exposes the balance of the token to the ISP to allow verifying that it is sufficient for retrieving the specified number of incentives. After the protocol, the user has an updated token $\tau'_{\mathcal{I}}$ with the balance w' that should be equal to $w - v$. Again, the ISP obtains a double-spending tag dstag *to identify cheating users and* hid *is for the definition only.*

Querier protocols

$((\text{regQ}_{qid}, st_{qid}), b_{\mathcal{RA}}) \leftarrow \texttt{RegisterQ} \left\langle \mathcal{Q}(\text{pk}_{\mathcal{RA}}, qid), \mathcal{RA}(\text{pk}_{\mathcal{RA}}, \text{sk}_{\mathcal{RA}}) \right\rangle$

This protocol allows a querier to register for a specific query identity qid. The querier obtains the querier registration value regQ_{qid} *and subscription token* st_{qid}*, which is required to collect reports from the SP.*

$(\mathfrak{M}, b_{\mathcal{SP}}) \leftarrow \texttt{CollectReports} \left\langle \mathcal{Q}(\text{CRS}, \text{pk}_{\mathcal{RA}}, qid, \text{regQ}_{qid}, st_{qid}), \mathcal{SP}(\text{pk}_{\mathcal{RA}}, \mathfrak{C}) \right\rangle$

This protocol is used by the querier to collect all the reports for a subscribed query identity qid from the SP. Hereby, $\mathfrak{C}$ is a set of reports together with their public bulletin information of the form $(c, com_{\mathcal{V}})$. The querier outputs a set $\mathfrak{M}$ containing the decoded data reports together with their bulletin commitment, ie. tuples of the form $(m, com_{\mathcal{V}}, \text{k}_{\text{mac}})$.

$(r_{\mathcal{V}}, (com_{\mathcal{V}}, v, r_{\mathcal{V}}, q)) \leftarrow \texttt{PostIncentives} \langle \mathcal{Q}(\text{CRS}, v, com_{\mathcal{V}}, q), \mathcal{I} \rangle$

This protocol allows the querier to post v incentive points as a reward for a submitted report on the bulletin board. $r_{\mathcal{V}}$ is a random nonce required to verify its delivery. q is an indicator for the quality of the data report corresponding to $com_{\mathcal{V}}$ as it has been perceived by the querier.

$(b_{\mathcal{Q}}, b_{\mathcal{I}}) \leftarrow \texttt{VerifyDelivery} \left\langle \mathcal{Q}(com_{\mathcal{V}}, \text{k}_{\text{mac}}, r_{\mathcal{V}}), \mathcal{I}(com_{\mathcal{V}}, t_{\text{mac}}) \right\rangle$

This protocol checks if t_{mac} is a valid proof of delivery for the incentive points posted for the report identified by $\text{com}_{\mathcal{V}}$ *with the verification key* k_{mac}*. In this case, $b_{\mathcal{Q}} = 1$.*

Double-spending detection and token verification algorithms

$(\text{pk}_{\mathcal{U}}, \Pi)$ or $\perp \leftarrow \texttt{IdentDS}(\text{pk}_{\mathcal{I}}, \text{dstag}_0, \text{dstag}_1)$

Given two double-spending tags dstag_0 *and* dstag_1*, this algorithm outputs the public key of a user*

if the double-spending tags originate from this user using the same balance token twice to collect or redeem incentives. Moreover, it outputs a proof Π that allows verifying that the owner of $\mathsf{pk}_{\mathcal{U}}$ is indeed guilty of double-spending. The same algorithm can be used to check double-spending for counter tokens if $\mathsf{pk}_{\mathcal{I}}$ is replaced with $\mathsf{pk}_{\mathcal{SP}}$.

$b \leftarrow \mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi)$

This algorithm allows verifying a proof Π to see if the user identified by $\mathsf{pk}_{\mathcal{U}}$ is guilty of double-spending incentive tokens. For counter tokens, $\mathsf{pk}_{\mathcal{I}}$ has to be replaced with $\mathsf{pk}_{\mathcal{SP}}$.

$\{0,1\} \leftarrow \mathtt{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{SP}}, ctr)$

The report counter token verification algorithm returns 1 if $\tau_{\mathcal{SP}}$ is a valid report counter token with the counter value ctr for the query identity qid belonging to the user identified by $\mathsf{pk}_{\mathcal{U}}$.

$\{0,1\} \leftarrow \mathtt{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w)$

The balance token verification algorithm returns 1 if $\tau_{\mathcal{I}}$ is a valid balance token with the balance w belonging to the user identified by $\mathsf{pk}_{\mathcal{U}}$ and 0 otherwise.

Correctness

An instantiation GI *of I3PS is called* correct, *if the following properties hold for all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$, $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(1^n)$, ISP key-pairs $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$, user key-pairs $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$ and parties $\mathcal{U}$, $\mathcal{SP}$ and $\mathcal{I}$ honestly following the protocols*

Correctness of balance token issuing

For all outputs of the issue protocol $((\tau_{\mathcal{I}}, b_{\mathcal{U}}), b_{\mathcal{I}}) \xleftarrow{\$} \mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \right\rangle$, it holds that

$$b_{\mathcal{U}} \stackrel{?}{=} b_{\mathcal{I}} \stackrel{?}{=} 1 \quad \wedge \quad \mathtt{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, 0) \stackrel{?}{=} 1$$

Correctness report counter token issuing

For all query identities qid in the query identity space, report limits $max_{qid} \in \mathbb{Z}_p$ it holds that

$$((\tau_{\mathcal{SP}}, 1), 1) \leftarrow \mathtt{RegisterUser}\left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \\ \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \end{matrix} \right\rangle$$
$$\wedge\ \mathtt{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{SP}}, max_{qid}) \stackrel{?}{=} 1$$

Correctness of data reporting

For all query identities qid and for all report counter tokens $\tau_{\mathcal{SP}}$ for qid, where $\mathtt{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{SP}}, ctr) \stackrel{?}{=} 1$ for a counter value $ctr > 0$, querier registration values and subscription tokens

$$((\mathrm{regQ}_{qid}, st_{qid}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterQ}\left\langle \mathcal{Q}(\mathsf{pk}_{\mathcal{RA}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \right\rangle$$

and sets $\mathfrak{C}$ *containing tuples* $(\widehat{c}, \widehat{com}_{\mathcal{V}})$ *of reports and public bulletin commitments, it holds that*

$$\begin{pmatrix}(\tau_{\mathcal{V}}, \tau^*_{\mathcal{SP}}, 1),\\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, 1)\end{pmatrix} \leftarrow \mathtt{SubmitReport}\left\langle\begin{matrix}\mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}, m),\\ \mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})\end{matrix}\right\rangle$$
$$\wedge\ \mathtt{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*_{\mathcal{SP}}, ctr - 1) \stackrel{?}{=} 1$$
$$\wedge\ (\mathfrak{M}, 1) \leftarrow \mathtt{CollectReports}\left\langle\begin{matrix}\mathcal{Q}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, qid, \mathsf{regQ}_{qid}, st_{qid}),\\ \mathcal{SP}(\mathsf{pk}_{\mathcal{RA}}, \mathfrak{C} \cup \{(c, com_{\mathcal{V}})\})\end{matrix}\right\rangle$$
$$\wedge\ (m, com_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}}) \in \mathfrak{M}$$

where $\mathsf{k}_{\mathrm{mac}}$ *is the same message authentication key as included in* $\tau_{\mathcal{V}}$*.*

Correctness of incentive collection

For all tokens $\tau_{\mathcal{I}}$*, balances* $w \in \mathbb{Z}_p$ *with* $\mathtt{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w) \stackrel{?}{=} 1$*, values* $v \in \mathbb{Z}_p$*, quality indicators* q *from the quality space and* $(\tau_{\mathcal{V}}, com_{\mathcal{V}})$ *pairs generated by a* `SubmitReport` *run protocol between* $\mathcal{U}$ *and* $\mathcal{SP}$ *it holds that*

$$(r_{\mathcal{V}}, (com_{\mathcal{V}}, v, r_{\mathcal{V}}, q)) \leftarrow \mathtt{PostIncentives}\,\langle \mathcal{Q}(\mathsf{CRS}, v, com_{\mathcal{V}}, q), \mathcal{I}\rangle$$
$$\wedge\ ((\tau^*_{\mathcal{I}}, 1, w^*), (t_{\mathrm{mac}}, \mathsf{dstag}, \mathsf{hid}, 1)) \leftarrow \mathtt{Collect}\left\langle\begin{matrix}\mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, \tau_{\mathcal{V}}, m),\\ \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v)\end{matrix}\right\rangle$$
$$\wedge\ \mathtt{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*_{\mathcal{I}}, w^*) \stackrel{?}{=} 1$$
$$\wedge\ w^* \stackrel{?}{=} w + v$$

Correctness of incentieve redemption

For all tokens $\tau_{\mathcal{I}}$*, balances* $w \in \mathbb{Z}_p$ *with* $\mathtt{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w) \stackrel{?}{=} 1$ *and values* $v \in \mathbb{Z}_p$*, we have that*

$$((\tau'_{\mathcal{I}}, 1, w'), (\mathsf{dstag}, \mathsf{hid}, 1)) \leftarrow \mathtt{Redeem}\,\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, v), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v)\rangle$$
$$\wedge\ \mathtt{UVer}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau'_{\mathcal{I}}, w - v) \stackrel{?}{=} 1$$

Correctness of delivery verification

For all $(\tau_{\mathcal{V}}, com_{\mathcal{V}})$ *generated by* `SubmitReport`*, all* $r_{\mathcal{V}}$ *generated by* `PostIncentives` *protocol runs for* $com_{\mathcal{V}}$ *and all tags* t_{mac} *generated by the corresponding* `Collect` *calls for* $com_{\mathcal{V}}$ *and* $\tau_{\mathcal{V}}$ *it holds that*

$$(1, 1) \leftarrow \mathtt{VerifyDelivery}\,\langle \mathcal{Q}(com_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}}), \mathcal{I}(com_{\mathcal{V}}, t_{\mathrm{mac}})\rangle$$

where $\mathsf{k}_{\mathrm{mac}}$ *is the same message authentication key as included in* $\tau_{\mathcal{V}}$*.*

7.4 Security

In this section, we define the security notions for our model. The notions are derived from PEPSICo, BBA+ and the previous model. We want to preserve the security properties from the underlying models and ensure that our modifications work as intended. Therefore, the used reporting mechanism (encryption, subscription matching and decryption of reports) should preserve the *node privacy* and *query privacy* notions of PEPSICo while both of the used BBA+ instantiations, in the incentive and the report limitation mechanism, should provide it's security properties. The exception is the *user privacy* property, which has to be combined with the *report unlinkability* property of PEPSICo as within the interim model.

To avoid having multiple similar security properties with the same name, we combine the security properties for the incentive and report limitation mechanism. An exception is the *balance-binding* property, where we have a slightly different goal for the report limitation mechanism. Instead of ensuring that the latest token always has the legitimately collected balance, we require that it is not possible to submit more reports than the allowed maximum. Therefore, we introduce a modified version of this notation that we call *limit-binding*. With the *owner-binding* property, we also address the security of the bulletin mechanism. Moreover, we specify the *verifiable delivery* property to ensure the soundness of the `VerifyDelivery` algorithm.

When we talk about successful protocol runs within the security definitions, this means that the run has been accepted by the SP or ISP. The view of a party during a protocol run consists of all its inputs, outputs, messages sent and messages received.

Within the security experiments we define in this chapter, the adversary has access to specific oracles that allow the interaction with the experiment. In Figures 7.2 and 7.3, we define some of the oracles upfront, as they are used within multiple experiments. Within these oracles, the adversary $\mathcal{A}$ impersonates the user. The oracles use the following global variables to store the state of the system operators

$\mathfrak{I}$	Set of $\mathsf{pk}_{\mathcal{U}}$s which already have a balance token issued
$balance_{\mathsf{pk}'_{\mathcal{U}}}$	Current legitimately accumulated balance of the user identified by $\mathsf{pk}'_{\mathcal{U}}$
$\mathfrak{R}$	Set of tuples $(\mathsf{pk}_{\mathcal{U}}, qid)$ where $\mathsf{pk}_{\mathcal{U}}$ is already registered for qid
$\mathfrak{L}(qid)$	Previously defined max_{qid}, this is used to allow $\mathcal{A}$ to choose max_{qid} but additionally enforcing that it remains consistent per query identity
$\mathfrak{C}$	Set of tuples of submitted reports $(c, com_{\mathcal{V}})$ stored by $\mathcal{SP}$
$\mathfrak{S}$	Set of tuples $(\mathsf{pk}'_{\mathcal{U}}, com_{\mathcal{V}})$ corresponding to reports submitted by $\mathcal{A}$ where the user identity $\mathsf{pk}'_{\mathcal{U}}$ could be extracted from the interaction
$ctr_{pk'_{\mathcal{U}}}^{\mathsf{regMN}_{qid}}$	Counter of the number of reports the user identified by $\mathsf{pk}'_{\mathcal{U}}$ submitted for qid, where regMN_{qid} is the corresponding mobile node registration value contained in the report c. These counters have to be initialized with 0
$\mathfrak{P}$	Set of tuples $(com_{\mathcal{V}}, v, r_{\mathcal{V}})$ that have been posted to the bulletin board

$\mathfrak{V}$	Set of tuples $(\mathsf{pk}'_{\mathcal{U}}, com_{\mathcal{V}})$ where the user identified by $\mathsf{pk}'_{\mathcal{U}}$ has successfully collected incentives posted for $com_{\mathcal{V}}$
$\mathfrak{E}$	Set of all user identities $\mathsf{pk}_{\mathcal{U}}$ for which there has been a successful call to MalRedeem
$\mathfrak{CI}_Q$	Set of all query identities for which $\mathcal{A}$ is registered as a querier

MalIssue

This oracle allows $\mathcal{A}$ to obtain a balance token by invoking the `Issue` protocol with the ISP played by the experiment. The oracle verifies that the user ID $\mathsf{pk}_{\mathcal{U}}$ used by $\mathcal{A}$ has not been used in a successful MalIssue call before and initiates $balance_{\mathsf{pk}_{\mathcal{U}}}$ with 0.

MalRegisterUser

This oracle allows $\mathcal{A}$ to register as a user identified by $\mathsf{pk}_{\mathcal{U}}$ for the query identity qid by invoking the `RegisterUser` protocol. The report limit max_{qid} for this query identity can be chosen by the adversary but has to be consistent per query identity, which is validated by the oracle. Moreover, the oracle checks that $\mathsf{pk}_{\mathcal{U}}$ has not been registered for qid before.

MalSubmitReport

$\mathcal{A}$ can use this oracle to submit reports for a query identity qid by invoking `SubmitReport`. As `SubmitReport` is not identifying, the oracle makes use of the trapdoor `td` to link the transaction to a user identity $\mathsf{pk}'_{\mathcal{U}}$. Moreover, the oracle keeps track of the number of submitted reports for this $(\mathsf{pk}'_{\mathcal{U}}, qid)$ pair with the counter $ctr_{pk'_{\mathcal{U}}}^{\mathsf{regMN}_{qid}}$. The oracle uses the mobile node registration value regMN_{qid} as it can be extracted from the submitted report c and is suited as it is also used to match the report to querier subscriptions later.

MalCollect

Using this oracle, $\mathcal{A}$ can collect all the incentives that have been posted for a bulletin commitment $com_{\mathcal{V}}$. For each of the posted incentives, the oracle invokes the `Collect` protocol with $\mathcal{A}$ and if the transaction was successful, uses the trapdoor `td` to extract the user ID used by $\mathcal{A}$ during the interaction and updates $balance_{\mathsf{pk}'_{\mathcal{U}}}$ accordingly.

MalRedeem

With this oracle, $\mathcal{A}$ can redeem some of the balance previously accumulated on a user's balance token. Therefore, the oracle invokes the `Redeem` protocol with $\mathcal{A}$ for the balance w and redemption value v and, if the protocol run was successful, extracts the user identity $\mathsf{pk}'_{\mathcal{U}}$ that $\mathcal{A}$ used during the protocol run and updates $balance_{\mathsf{pk}'_{\mathcal{U}}}$ accordingly.

MalRegisterQ

This oracle allows $\mathcal{A}$ to register as a querier for a query identity qid.

MalPostIncentives

Whenever $\mathcal{A}$ calls this oracle, an honest querier is created that collects all the reports previously

submitted for *qid* and post incentives for them. This allows $\mathcal{A}$ to collect these incentives later using MalCollect.

7.4.1 Data-hiding

This property adapts the *node privacy* property of PEPSICo to our model. It demands the confidentiality of data reports against the SP, unauthorized queriers and other users, even if they all collude. It is modeled as the indistinguishability of data reports generated using two query identity/message pairs chosen by the adversary. The adversary is not allowed to register for the challenge query identities, neither as user nor as querier. However, if the SP is not corrupted, The adversary can have reports for any query identity be submitted by an honest user and collect any submitted messages for which they can provide a valid subscription token.

For this property, we additionally define the oracles CorruptSP, SubmitReport and CollectReports (Figure 7.4).

CorruptSP
: This oracle allows $\mathcal{A}$ to corrupt the SP. It marks the SP to be corrupted and returns $\mathsf{sk}_{\mathcal{SP}}$.

SubmitReport
: When this oracle is called by $\mathcal{A}$ a new user identity $\mathsf{pk}_{\mathcal{U}}$ is created and registered for *qid*. Afterward, the user identified by $\mathsf{pk}_{\mathcal{U}}$ submits a report containing the report data m which is stored in the list of submitted reports. This oracle does not return any information to $\mathcal{A}$.

CollectReports
: When this oracle is called, a new querier is registered for the query identity *qid* and uses the subscription tokens specified by the adversary to try to collect all reports that have previously been submitted for qid. A list of the obtained information (report data m, bulletin commitment $com_{\mathcal{V}}$ and message authentication key $\mathsf{k}_{\mathrm{mac}}$) is returned to $\mathcal{A}$.

Definition 7.2 (Data-hiding) *An instantiation* GI *of our model is* data-hiding*, if no adversary can win the experiment* $\mathrm{Exp}^{\mathrm{DH}}_{\mathrm{GI},\mathcal{A}}$ *with more than negligible advantage, ie. for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\mathsf{Adv}^{\mathrm{DH}}_{\mathrm{GI},\mathcal{A}}(n) := \left| \Pr\left[\mathrm{Exp}^{\mathrm{DH}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1 \right] - \frac{1}{2} \right| \leq \mathsf{negl}(n)$$

7.4.2 Subscription-hiding

Our adaption of PEPSICo's *query privacy* notion demands that the query identity corresponding to a query subscription is hidden from the SP, as well as queriers and users that are not registered for the same query identity, even if they collude with the SP.

MalIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{I}$ **then**
  $((\tau_{\mathcal{I}}, b_{\mathcal{U}}), b_{\mathcal{I}}) \leftarrow \mathtt{Issue}\left\langle \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$
  if $b_{\mathcal{I}} \stackrel{?}{=} 1$ **then**
    $\mathfrak{I} \leftarrow \mathfrak{I} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
    $balance_{\mathsf{pk}_{\mathcal{U}}} := 0$
  endif
endif

MalRegisterUser($\mathsf{pk}_{\mathcal{U}}, qid, max_{qid}$)

if $\{(\mathsf{pk}_{\mathcal{U}}, qid)\} \notin \mathfrak{R}$ **then**
  if $max_{qid} > 0$ and $(\mathfrak{L}(qid) \stackrel{?}{=} \perp$ or $\mathfrak{L}(qid) \stackrel{?}{=} max_{qid})$ **then**
    $\mathfrak{L}(qid) := max_{qid}$
    $((\tau_{\mathcal{SP}}, b_{\mathcal{U}}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterUser}\left\langle \begin{matrix} \mathcal{A}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \\ \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \end{matrix}\right\rangle$
    if $b_{\mathcal{RA}} \stackrel{?}{=} 1$ **then**
      $\mathfrak{R} := \mathfrak{R} \cup \{(\mathsf{pk}_{\mathcal{U}}, qid)\}$
    endif
  endif
endif

MalSubmitReport(qid, m)

$\left\langle \begin{matrix} (\tau_{\mathcal{V}}, \tau^{*}_{\mathcal{SP}}, b_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}}) \end{matrix}\right\rangle \leftarrow \mathtt{SubmitReport}\left\langle \begin{matrix} \mathcal{A}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}, m), \\ \mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix}\right\rangle$

if $b_{\mathcal{I}} \stackrel{?}{=} 1$ **then**
  $\mathfrak{C} := \mathfrak{C} \cup \{(c, com_{\mathcal{V}})\}$
  $pk'_{\mathcal{U}} := \mathtt{ExtractUID}(\mathsf{td}, \mathsf{hid})$
  $\mathfrak{S} := \mathfrak{S} \cup \{(pk'_{\mathcal{U}}, com_{\mathcal{V}})\}$
  $(\mathsf{regMN}_{qid}, c_1) := c$
  $ctr^{\mathsf{regMN}_{qid}}_{\mathsf{pk}'_{\mathcal{U}}} := ctr^{\mathsf{regMN}_{qid}}_{\mathsf{pk}'_{\mathcal{U}}} + 1$
endif

Figure 7.2: Oracle definitions for the security notation of I3PS, part 1

MalCollect($com_{\mathcal{V}}$)

for $(com^*_{\mathcal{V}}, v, r_{\mathcal{V}}) \in \mathfrak{P}$ where $com^*_{\mathcal{V}} \stackrel{?}{=} com_{\mathcal{V}}$ **do**
  $((\tau^*_{\mathcal{I}}, b_{\mathcal{U}}, w^*), (t_{\text{mac}}, \text{dstag}, \text{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Collect} \left\langle \begin{matrix} \mathcal{A}(\text{pk}_{\mathcal{I}}, \text{pk}_{\mathcal{U}}, \text{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, \tau_{\mathcal{V}}, m), \\ \mathcal{I}(\text{pk}_{\mathcal{I}}, \text{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v) \end{matrix} \right\rangle$
  if $b_{\mathcal{I}} \stackrel{?}{=} 1$ **then**
    $pk'_{\mathcal{U}} := \texttt{ExtractUID}(\text{td}, \text{hid})$
    $\mathfrak{V} := \mathfrak{V} \cup \{(pk'_{\mathcal{U}}, com_{\mathcal{V}})\}$
    $balance_{\text{pk}'_{\mathcal{U}}} := balance_{\text{pk}'_{\mathcal{U}}} + v$
  endif
endfor

MalRedeem(w, v)

$((\tau'_{\mathcal{I}}, b_{\mathcal{U}}, w'), (\text{dstag}, \text{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Redeem} \left\langle \mathcal{A}(\text{pk}_{\mathcal{I}}, \text{pk}_{\mathcal{U}}, \text{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, v), \mathcal{I}(\text{pk}_{\mathcal{I}}, \text{sk}_{\mathcal{I}}, w, v) \right\rangle$
if $b_{\mathcal{I}} \stackrel{?}{=} 1$ **then**
  $\text{pk}'_{\mathcal{U}} := \texttt{ExtractUID}(\text{td}, \text{hid})$
  $\mathfrak{E} := \mathfrak{E} \cup \{pk'_{\mathcal{U}}\}$
  $balance_{\text{pk}'_{\mathcal{U}}} := balance_{\text{pk}'_{\mathcal{U}}} - v$
endif

MalRegisterQ(qid)

$((\text{regQ}_{qid}, st_{qid}), b_{\mathcal{RA}}) \leftarrow \texttt{RegisterQ} \left\langle \mathcal{A}(\text{pk}_{\mathcal{RA}}, qid), \mathcal{RA}(\text{pk}_{\mathcal{RA}}, \text{sk}_{\mathcal{RA}}) \right\rangle$
$\mathfrak{CI}_Q := \mathfrak{CI}_Q \cup \{(qid)\}$

MalPostIncentives(qid)

$((\text{regQ}_{qid}, st_{qid}), b_{\mathcal{RA}}) \leftarrow \texttt{RegisterQ} \left\langle \mathcal{Q}(\text{pk}_{\mathcal{RA}}, qid), \mathcal{RA}(\text{pk}_{\mathcal{RA}}, \text{sk}_{\mathcal{RA}}) \right\rangle$
$(\mathfrak{M}, b_{\mathcal{SP}}) \leftarrow \texttt{CollectReports} \left\langle \mathcal{Q}(\text{CRS}, \text{pk}_{\mathcal{RA}}, qid, \text{regQ}_{qid}, st_{qid}), \mathcal{SP}(\text{pk}_{\mathcal{RA}}, \mathfrak{C}) \right\rangle$
for $(m, com_{\mathcal{V}}, \text{k}_{\text{mac}}) \in \mathfrak{M}$ **do**
  $v := 1$
  $q := 1$
  $(r_{\mathcal{V}}, (com_{\mathcal{V}}, v, r_{\mathcal{V}}, q)) \leftarrow \texttt{PostIncentives} \langle \mathcal{Q}(\text{CRS}, v, com_{\mathcal{V}}, q), \mathcal{I} \rangle$
  $\mathfrak{P} := \mathfrak{P} \cup \{(com_{\mathcal{V}}, v, r_{\mathcal{V}})\}$
endfor

Figure 7.3: Oracle definitions for the security notation of I3PS, part 2

CorruptSP

$corruptSP := 1$

return $\mathsf{sk}_{\mathcal{SP}}$

SubmitReport(qid, m))

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$

$max_{qid} := 1$

$\mathfrak{L}(qid) := 1$

$((\tau_{\mathcal{SP}}, b_{\mathcal{U}}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterUser} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \\ \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \end{matrix} \right\rangle$

$\left(\begin{matrix} (\tau_{\mathcal{V}}, \tau^*_{\mathcal{SP}}, b_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}}) \end{matrix} \right) \leftarrow \mathtt{SubmitReport} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}, m), \\ \mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix} \right\rangle$

$\mathfrak{C} := \mathfrak{C} \cup \{(c, com_{\mathcal{V}})\}$

CollectReports$(qid, \vec{st})$

$((\mathsf{regQ}_{qid}, st^*_{qid}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterQ} \left\langle \mathcal{Q}(\mathsf{pk}_{\mathcal{RA}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \right\rangle$

for st **in** $\vec{st}$ **do**

  $(\mathfrak{M}, b_{\mathcal{SP}}) \leftarrow \mathtt{CollectReports} \left\langle \mathcal{Q}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, qid, \mathsf{regQ}_{qid}, st), \mathcal{SP}(\mathsf{pk}_{\mathcal{RA}}, \mathfrak{C}) \right\rangle$

endfor

return $\mathfrak{M}$

Figure 7.4: CorruptSP, SubmitReport and CollectReports oracles for the *data-hiding* experiment

$\mathrm{Exp}^{\mathrm{DH}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$

$((qid_0, m_0), (qid_1, m_1), state_0) \leftarrow \mathcal{A}_0^{\text{CorruptSP,MalRegisterUser,MalRegisterQ,SubmitReport,CollectReports}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}})$

$b \leftarrow \{0,1\}$

$max_{qid_b} := 1$

$((\tau_{\mathcal{SP}}, b_{\mathcal{U}}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterUser} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid_b), \\ \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid_b, max_{qid_b}) \end{matrix} \right\rangle$

if $corruptSP \stackrel{?}{=} 1$ **do**

$\left(\begin{matrix} (\tau_{\mathcal{V}}, \tau^*_{\mathcal{SP}}, b'_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{state}_1) \end{matrix} \right) \leftarrow \mathtt{SubmitReport} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid_b, \tau_{\mathcal{SP}}, m_b), \\ \mathcal{A}_1^{\text{MalRegisterUser,MalRegisterQ,SubmitReport,CollectReports}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, state_0) \end{matrix} \right\rangle$

if $b'_{\mathcal{U}} \stackrel{?}{=} 0$ **then**

return 0

endif

$b' \leftarrow \mathcal{A}_2^{\text{MalRegisterUser,MalRegisterQ,SubmitReport,CollectReports}}(state_1)$

else

$\left(\begin{matrix} (\tau_{\mathcal{V}}, \tau^*_{\mathcal{SP}}, b'_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}}) \end{matrix} \right) \leftarrow \mathtt{SubmitReport} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid_b, \tau_{\mathcal{SP}}, m_b), \\ \mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix} \right\rangle$

$\widehat{\mathfrak{C}} := \{(c, com_{\mathcal{V}})\}$

$(b', b_{\mathcal{SP}}) \leftarrow \mathtt{CollectReports}^* \left\langle \begin{matrix} \mathcal{A}_1^{\text{CorruptSP,MalRegisterUser,MalRegisterQ,SubmitReport,CollectReports}}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, state_0), \\ \mathcal{SP}(\mathsf{pk}_{\mathcal{RA}}, \widehat{\mathfrak{C}}) \end{matrix} \right\rangle$

endif

The experiment returns 1 iff all of the following conditions are met:

- $b \stackrel{?}{=} b'$
- $\{qid_0, qid_1\} \cap \mathfrak{CI}_Q \stackrel{?}{=} \emptyset$
- $\nexists \mathsf{pk}_{\mathcal{U}} : ((\mathsf{pk}_{\mathcal{U}}, qid_0) \in \mathfrak{R} \vee (\mathsf{pk}_{\mathcal{U}}, qid_1) \in \mathfrak{R})$
- If $corruptSP \stackrel{?}{=} 1$, then $\mathcal{A}$ did not query SubmitReport for qid_0 or qid_1.

Figure 7.5: *Data-hiding* experiment for I3PS. The oracles available to $\mathcal{A}$ are defined in the Figures 7.2 to 7.4. In the `CollectReports`* protocol, $\mathcal{SP}$ behaves as it would in the `CollectReports` protocol. However, $\mathcal{A}$ has to adapt his behavior in any case to the modified input and output on his side. For example, $\mathcal{A}$ could try to find regQ_{qid_b} and a corresponding subscription token. Moreover, instead of outputting a message list, $\mathcal{A}$ outputs a guess for b.

In our definition of the *subscription-hiding* experiment, the adversary only has the oracles required to interact with the RA. In opposite to the previous experiment, the usage of the other oracles available to the adversary in the original PEPSICo game is never allowed for the two challenge query identities. However, the adversary can use the corruption oracles to obtain the registration values any other query identity, allowing him to compute the corresponding algorithms on his own.

$\mathrm{Exp}^{\mathrm{SH}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(qid_0, qid_1, state_0) \leftarrow \mathcal{A}_0^{\mathrm{MalRegisterUser,MalRegisterQ}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$

$b \leftarrow \{0,1\}$

$((\mathsf{regQ}_{qid_b}, st_{qid_b}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterQ}\left\langle \mathcal{Q}(\mathsf{pk}_{\mathcal{RA}}, qid_b), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \right\rangle$

$(\mathfrak{M}, b') \leftarrow \mathtt{CollectReports}\left\langle \mathcal{Q}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, qid_b, \mathsf{regQ}_{qid_b}, st_{qid_b}), \mathcal{A}_1^{\mathrm{MalRegisterUser,MalRegisterQ}}(\mathsf{pk}_{\mathcal{RA}}, \emptyset, state_0) \right\rangle$

The experiment returns 1 iff all of the following conditions are met:

- $b \stackrel{?}{=} b'$
- $\{qid_0, qid_1\} \cap \mathfrak{CI}_Q \stackrel{?}{=} \emptyset$
- $\nexists \mathsf{pk}_{\mathcal{U}} : ((\mathsf{pk}_{\mathcal{U}}, qid_0) \in \mathfrak{R} \vee (\mathsf{pk}_{\mathcal{U}}, qid_1) \in \mathfrak{R})$

Figure 7.6: *Subscription-hiding* experiment for I3PS. The oracles available to $\mathcal{A}$ are defined in Figure 7.2 and Figure 7.3

Definition 7.3 (Subscription-hiding) *An instantiation* GI *of our model is called* subscription-hiding *if no adversary can win the query privacy experiment* $\mathrm{Exp}^{\mathrm{SH}}_{\mathrm{GI},\mathcal{A}}$ *with more than negligible advantage, ie. for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\mathrm{Adv}^{\mathrm{SH}}_{\mathrm{GI},\mathcal{A}}(n) := \left| \Pr\left[\mathrm{Exp}^{\mathrm{SH}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1 \right] - \frac{1}{2} \right| \leq \mathsf{negl}(n)$$

7.4.3 Trapdoor-linkability

The *trapdoor-linkability* property addresses the issue that the formalization of some of the security properties we want to achieve requires to link each transaction with a user and token. This conflicts with our demand that transactions are anonymous and unlinkable. Therefore, privacy can be abolished given a trapdoor which should be kept secret by the TTP.

Definition 7.4 (Trapdoor-linkability) *For a fixed security parameter n and CRS* CRS*, let* $\mathfrak{V}^{\mathtt{SubmitReport}}_{n,\mathrm{CRS}}$ *be the set of all views of the SP on successful* `SubmitReport` *protocol runs with any (possibly malicious) party and any* $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}$*. Hereby, a view is of the following form:*

$$\mathsf{view} := (\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{msgs}, c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{SP}})$$

where msgs *is the bit string of all exchanged messages during the protocol run. Similar, let* $\mathfrak{V}^{\mathtt{Collect}}_{n,\mathrm{CRS}}$ *be the*

set of all views of the ISP on successful Collect *protocol runs with any party and any* $\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}$. *Hereby, a view is of the form*

$$\mathsf{view} := (\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v, \mathsf{msgs}, t_{\mathrm{mac}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})$$

and $\mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{Redeem}}$ *be the set of all views of the ISP on successful* Redeem *protocol runs with any party and any* $\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}$ *where view is of the form*

$$\mathsf{view} := (\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v, \mathsf{msgs}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})$$

An instantiation GI *of our model is* trapdoor-linkable *if the following two conditions hold*

Completness

For all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$ *and* $\mathsf{view} \in \mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{SubmitReport}}$, *containing a hidden user ID* hid, *there exist inputs* $\mathsf{pk}_{\mathcal{RA}}$, $\mathsf{pk}_{\mathcal{U}}$, $\mathsf{sk}_{\mathcal{U}}$, *qid*, $\tau_{\mathcal{SP}}$, *m and random choices for an honest user* $\mathcal{U}$ *and honest ISP* $\mathcal{I}$ *such that a* SubmitReport *protocol run between* $\mathcal{U}$ *and* $\mathcal{I}$ *with these inputs leads to a view* $\mathsf{view}' \in \mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{SubmitReport}}$ *containing the same hidden user ID* hid *as in* view.

Analogously, for all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$ *and* $\mathsf{view} \in \mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{Collect}}$, *containing the hidden user ID* hid, *there exists inputs* $\mathsf{pk}_{\mathcal{U}}$, $\mathsf{sk}_{\mathcal{U}}$, $\tau_{\mathcal{I}}$, w, $\tau_{\mathcal{V}}$, m *and random choices for an honest user* $\mathcal{U}$ *and honest ISP* $\mathcal{I}$ *such that a* Collect *protocol run between* $\mathcal{U}$ *and* $\mathcal{I}$ *with these inputs leads to a view* $\mathsf{view}' \in \mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{Collect}}$ *containing the same hidden user ID* hid *as in* view.

Furthermore, for all $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$ *and* $\mathsf{view} \in \mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{Redeem}}$, *containing the hidden user ID* hid, *there exists inputs* $\mathsf{pk}_{\mathcal{U}}$, $\mathsf{sk}_{\mathcal{U}}$, $\tau_{\mathcal{I}}$, w, v *and random choices for an honest user* $\mathcal{U}$ *and honest ISP* $\mathcal{I}$ *such that a* Redeem *protocol run between* $\mathcal{U}$ *and* $\mathcal{I}$ *with these inputs leads to a view* $\mathsf{view}' \in \mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{Redeem}}$ *containing the same hidden user ID* hid *as in* view.

Extractability

There exists a PPT *algorithm* ExtractUID *such that for any* $n \in \mathbb{N}$, $(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$, *and* $\mathsf{view} := (\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{msgs}, c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{SP}}) \in \mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{SubmitReport}}$ *of a* SubmitReport *protocol run with an honest user on input* $\mathsf{pk}_{\mathcal{U}}$, $\mathtt{ExtractUID}(\mathsf{td}, \mathsf{hid})$ *outputs* $\mathsf{pk}_{\mathcal{U}}$. *The same needs to hold for* ExtractUID *with respect to views from* $\mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{Collect}}$ *and* $\mathfrak{V}_{n,\mathrm{CRS}}^{\mathtt{Redeem}}$.

This property implies that any fixed view view *cannot result from interactions with different users.*

7.4.4 Owner-binding

This property requires that balance and report limitation tokens are bound to a specific owner. Moreover, we require that incentive points can only be collected by the same user that submitted the report.

This property is described by multiple experiments representing the ways an adversary may try to impersonate honest users. First, in the OB-issue experiment, the goal of the adversary is trick the ISP into issuing a balance or report limitation token for an honest and uncorrupted user to the adversary,

which does not know the user's secret key. Next, in the OB-submit and OB-col-red experiment, the adversary has to make the ISP accept a forged token that has not been legitimately issued by the ISP and RA, respectively. Last, in the OB-bulletin experiment, the adversary has to successfully collect incentive points for a different user identity as it was used to submit the report.

Definition 7.5 (Owner-binding) *An instantiation* GI *of our model is called* owner-binding *if no* PPT *adversary can win* $\mathrm{Exp}^{\text{OB-issue}}_{\mathrm{GI},\mathcal{A}}$, $\mathrm{Exp}^{\text{OB-submit}}_{\mathrm{GI},\mathcal{A}}$, $\mathrm{Exp}^{\text{OB-col-red}}_{\mathrm{GI},\mathcal{A}}$ *or* $\mathrm{Exp}^{\text{OB-bulletin}}_{\mathrm{GI},\mathcal{A}}$ *as defined in Figure 7.7 and Figure 7.8 with more than negligible probability. More precisely, for all* PPT *adversaries* $\mathcal{A}$

$$\Pr\left[\mathrm{Exp}^{\text{OB-issue}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1 \vee \mathrm{Exp}^{\text{OB-submit}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1 \vee \mathrm{Exp}^{\text{OB-col-red}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1 \vee \mathrm{Exp}^{\text{OB-bulletin}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

7.4.5 Limit-binding

This property is the equivalent of the balance-binding property of BBA+ for the report limitation mechanism. Instead of requiring that claimed balance is correct, we require that it is not possible to submit more than max_{qid} reports for qid.

Definition 7.6 (Limit-binding) *Let* $\mathrm{Exp}^{\mathrm{LB}}_{\mathrm{GI},\mathcal{A}}$ *be defined as in Figure 7.9. An instantiation* GI *of our model has a* limit-binding *report mechanism if for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathrm{Exp}^{\mathrm{LB}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

7.4.6 Balance-binding

This property is the balance-binding property of BBA+ for the incentive mechanism, adapted to the interface of the advanced model. This property ensures that as long as a token is only used once the claimed balance in the scope of the `Redeem` protocol always coincides with the sum of points allegedly collected with this token.

Definition 7.7 (Balance-binding) *An instantiation* GI *of our model is called* balance-binding *if for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathrm{Exp}^{\mathrm{BB}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

7.4.7 Double-spending detection

We want to make sure that no balance or report limitation token is used more than once within a transaction. As this is difficult to enforce, we make use of BBA+ capability to identify users that commit such double-spending.

This property ensures that two transactions leading to the same token version number s have always been initiated by the same user. Moreover, this user can be identified by `IdentDS`, resulting in a valid proof of guilt Π.

$\mathrm{Exp}^{\text{OB-issue}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\text{MalIssue,MalRegisterUser,MalSubmitReport,MalPostIncentives,MalCollect,MalRedeem}}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$

The experiment returns 1 iff $\mathcal{A}$ did a sucessful call to MalIssue or MalRegisterUser on input of the given public key $\mathsf{pk}_{\mathcal{U}}$, ie.

$$\mathsf{pk}_{\mathcal{U}} \in \mathfrak{I} \vee (\exists qid : (\mathsf{pk}_{\mathcal{U}}, qid) \in \mathfrak{R})$$

$\mathrm{Exp}^{\text{OB-submit}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\text{MalRegisterUser,MalSubmitReport}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}})$

The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalSubmitReport for an extracted public key $\mathsf{pk}'_{\mathcal{U}}$ for which there has been no successful execution of MalRegisterUser, ie. if there exists $\mathsf{pk}'_{\mathcal{U}}$ such that

$$(\exists com_{\mathcal{V}} : (\mathsf{pk}'_{\mathcal{U}}, com_{\mathcal{V}}) \in \mathfrak{S}) \wedge (\forall qid : (pk'_{\mathcal{U}}, qid) \notin \mathfrak{R})$$

$\mathrm{Exp}^{\text{OB-col-red}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\text{MalIssue,MalRegisterUser,MalSubmitReport,MalPostIncentives,MalCollect,MalRedeem}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{I}})$

The experiment returns 1 iff $\mathcal{A}$ did a successful call to MalCollect or MalRedeem for an extracted public key $\mathsf{pk}'_{\mathcal{U}}$ for which there has been no successful execution of MalIssue, ie. if there exists $\mathsf{pk}'_{\mathcal{U}} \notin \mathfrak{I}$ such that

$$(\exists com_{\mathcal{V}} : (\mathsf{pk}'_{\mathcal{U}}, com_{\mathcal{V}}) \in \mathfrak{V}) \vee \mathsf{pk}'_{\mathcal{U}} \in \mathfrak{E}$$

Figure 7.7: *Owner-binding* experiments for the `Issue`, `SubmitReport`, `Collect` and `Redeem` protocols. The oracles available to $\mathcal{A}$ are defined in Figure 7.2 and Figure 7.3

$\mathrm{Exp}^{\text{OB-bulletin}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\text{MalIssue,MalRegisterUser,MalSubmitReport,MalPostIncentives,MalCollect}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{I}})$

The experiment returns 1 if there exists a tuple $(pk'_{\mathcal{U}}, com_{\mathcal{V}})$ such that

$$(pk'_{\mathcal{U}}, com_{\mathcal{V}}) \in \mathfrak{V} \wedge (\mathsf{pk}'_{\mathcal{U}}, com_{\mathcal{V}}) \notin \mathfrak{S}$$

Figure 7.8: *Owner-binding* experiment for the bulletin mechanism. The oracles available to $\mathcal{A}$ are defined in Figure 7.2 and Figure 7.3.

$\mathrm{Exp}^{\mathrm{LB}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\text{MalRegisterUser,MalSubmitReport}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}})$

The experiment returns 1 iff all successful MalSubmitReport calls produced unique token version numbers and there exist a mobile node registration value regMN_{qid} and a user public key $\mathsf{pk}'_{\mathcal{U}}$ such that

$$ctr^{\mathsf{regMN}_{qid}}_{\mathsf{pk}'_{\mathcal{U}}} > \mathfrak{L}(qid)$$

provided that both variables have been defined during the run of the experiment.

Figure 7.9: *Limit-binding* experiment for I3PS. The oracles available to $\mathcal{A}$ are defined in Figure 7.2 and Figure 7.3.

$\mathrm{Exp}^{\mathrm{BB}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$

$b \leftarrow \mathcal{A}^{\text{MalIssue,MalRegisterUser,MalSubmitReport,MalPostIncentives,MalCollect}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{I}})$

Given that all successful MalIssue/MalCollect and MalRedeem calls produced unique token version numbers, the experiment returns 1 iff $\mathcal{A}$ did successful call MalRedeem for an extracted public key $\mathsf{pk}'_{\mathcal{U}}$ resulting in balance w^* that does not equal the sum of the collected/redeemed values, ie. $w^* \neq balance_{\mathsf{pk}'_{\mathcal{U}}}$.

Figure 7.10: *Balance-binding* experiment for I3PS

Definition 7.8 (Double-spending detection) *Let* $\mathrm{Exp}^{\mathrm{DSD}}_{\mathrm{GI},\mathcal{A}}$ *be defined as in Figure 7.11. An instantiation* GI *of our model ensures double-spending detection if there exists a negligible function* $\mathsf{negl}(n)$ *such that for all* PPT *adversaries* $\mathcal{A}$

$$\Pr\left[\mathrm{Exp}^{\mathrm{DSD}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

$\mathrm{Exp}^{\mathrm{DSD}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) \leftarrow \mathtt{IGen}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$
$b \leftarrow \mathcal{A}^{\mathrm{MalIssue,MalRegisterUser,MalSubmitReport,MalPostIncentives,MalCollect,MalRedeem}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$

The experiment returns 1 iff $\mathcal{A}$ did two successful MalCollect/MalRedeem or MalSubmitReport calls resulting in two views view_0 and view_1 including two double-spending tags $\mathsf{dstag}_0 = (s, z_0)$ and $\mathsf{dstag}_1 = (s, z_1)$ and extracted user public keys $\mathsf{pk}^{(0)}_{\mathcal{U}}$ and $\mathsf{pk}^{(1)}_{\mathcal{U}}$ (using `ExtractUID`) such that at least one of the following conditions is satisfied:

- $\mathsf{pk}^{(0)}_{\mathcal{U}} \neq \mathsf{pk}^{(1)}_{\mathcal{U}}$ or
- $\mathtt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1) \neq (\mathsf{pk}^{(0)}_{\mathcal{U}}, \Pi)$ or
- $\mathtt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1) \stackrel{?}{=} (\mathsf{pk}^{(0)}_{\mathcal{U}}, \Pi)$ but $\mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}^{(0)}_{\mathcal{U}}, \Pi) \stackrel{?}{=} 0$

Figure 7.11: *Double-spending detection* experiment for I3PS

7.4.8 Verifiable delivery

This property ensures the soundness of the `VerifyDelivery` algorithm. The ISP should not be able to falsely convince the querier of the successful delivery of incentives. The adversary can be a collusion of the SP and ISP together with several users and queriers. However, the queriers are not allowed to register for the query identity of the challenge report.

Definition 7.9 (Verifiable delivery) *An instantiation* GI *of our model has* verifiable delivery*, if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ *and for all* $(qid, m) \in \mathfrak{I} \times \mathfrak{M}$ *and the game* $\mathrm{Game}^{\mathrm{VD}}_{\mathrm{GI},\mathcal{A}}$ *as defined in Figure 7.12, there exists a negligible function* negl *such that*

$$\Pr\left[\mathrm{Game}^{\mathrm{VD}}_{\mathrm{GI},\mathcal{A}}(n, qid, m) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

7.4.9 Transaction unlinkability

As one of the main security goals of this model is to protect the privacy of the users, we require that no other party should be able to link transactions originating from the same user. Our definition of

$\mathrm{Game}^{\mathrm{VD}}_{\mathrm{GI},\mathcal{A}}(n, qid, m)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$

$max_{qid} := 1$

$\mathfrak{L}(qid) := 1$

$(\tau_{\mathcal{SP}}, b_{\mathcal{U}}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterUser} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \\ \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \end{matrix} \right\rangle$

$\begin{pmatrix} (\tau_{\mathcal{V}}, \tau^*_{\mathcal{SP}}, b'_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{state}_0) \end{pmatrix} \leftarrow \mathtt{SubmitReport} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}, m), \\ \mathcal{A}_0^{\mathrm{MalRegisterUser,MalRegisterQ}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix} \right\rangle$

if $b'_{\mathcal{U}} \stackrel{?}{=} 0$ **then**

  return 0

endif

$((\mathsf{regQ}_{qid}, st_{qid}), b_{\mathcal{RA}}) \leftarrow \mathtt{RegisterQ} \left\langle \mathcal{Q}(\mathsf{pk}_{\mathcal{RA}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \right\rangle$

$(\mathfrak{M}, \mathsf{state}_1) \leftarrow \mathtt{CollectReports} \left\langle \begin{matrix} \mathcal{Q}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, qid, \mathsf{regQ}_{qid}, st_{qid}), \\ \mathcal{A}_1^{\mathrm{MalRegisterUser,MalRegisterQ}}(\mathsf{pk}_{\mathcal{RA}}, \{(c, com_{\mathcal{V}})\}, \mathsf{state}_0) \end{matrix} \right\rangle$

$\mathsf{k}_{\mathrm{mac}} := \mathtt{getMacKey}(\tau_{\mathcal{V}})$

if $(m, com_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}}) \notin \mathfrak{M}$ **then**

  return 0

endif

$q := 1$

$(r_{\mathcal{V}}, (com_{\mathcal{V}}, v, r_{\mathcal{V}}, q, \mathsf{state}_2)) \leftarrow \mathtt{PostIncentives} \left\langle \mathcal{Q}(\mathsf{CRS}, v, com_{\mathcal{V}}, q), \mathcal{A}_2^{\mathrm{MalRegisterUser,MalRegisterQ}}(\mathsf{state}_1) \right\rangle$

$(b_{\mathcal{Q}}, b_{\mathcal{A}}) \leftarrow \mathtt{VerifyDelivery} \left\langle \mathcal{Q}(com_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}}, r_{\mathcal{V}}), \mathcal{A}_3^{\mathrm{MalRegisterUser,MalRegisterQ}}(com_{\mathcal{V}}, \mathsf{state}_2) \right\rangle$

return ($b_{\mathcal{Q}}$ and $qid \notin \mathfrak{CI}_Q$)

Figure 7.12: *Verifiable delivery* game for I3PS. Hereby, the `getMacKey` algorithm returns the message authentication key $\mathsf{k}_{\mathrm{mac}}$ contained in a bulletin token $\tau_{\mathcal{V}}$

transaction unlinkability is a combination of the *privacy* notion of BBA+ and the *report unlinkability* notion of PEPSICo, similar to the notion used within the interim model.

More precisely, our property demands that no collusion of the RA, the SP, and ISP, as well as users and queriers, cannot compromise the privacy of an honest user. Moreover, it ensures forward privacy similar to BBA+. In the *privacy* notion of BBA+, an adversary compromising a user can use its knowledge only to link the first transaction directly following the corruption to this user but all subsequent transactions are unaffected. We demand this for the `Collect` and `Redeem` protocols and, in addition, demand that, for all the query identities the user was registered to prior to his corruption, only the next `SubmitReport` call for each query identity can be linked. Therefore, even following a corruption of the user, the adversary can only link a small fraction of the user's transaction, limiting the potential impact.

For the definition, there are two security experiments which should be indistinguishable, the real-world experiment $\mathrm{Exp}^{\text{TU-real}}_{\text{GI},\mathcal{A}}$ and the ideal world experiment $\mathrm{Exp}^{\text{TU-ideal}}_{\text{GI},\mathcal{A}}$ (Figure 7.19). In both experiments, the adversary $\mathcal{A}$ plays the role of the SP and ISP and additionally knows the secret key of the RA. Moreover, $\mathcal{A}$ has access to various oracles for the interaction with honest users, which additionally can be corrupted. In the real world experiments, the user oracles behave in the same way an honest user would, whereas in the ideal world, the user code is replaced by a user simulator $\mathcal{U}_{\mathsf{Sim}}$ that does not have access to the personal user information that should not be disclosed within the interaction. $\mathcal{U}_{\mathsf{Sim}}$ does not have access to the user's secret key in any case but is allowed to know the user's public key for identifying transactions, such as `Issue` and `SubmitReport`.

In addition to simulating the protocols, the oracles have to ensure that certain conditions are met, ie. that users cannot register for the same query identity more than once and that they would not attempt to submit a report for a query identity they are not registered for. Additional, note that users that rejected a protocol run in the past (because they detected cheating by $\mathcal{A}$) also reject to participate in further interactions. To validate these conditions, the oracles make use of the following global variables to store the corresponding information:

$\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \ldots$	User specific global variables with the sub- or superscript $\mathcal{U}$ are used to store the private information of the corresponding user $\mathcal{U}$. An exception is the bit $b_{\mathcal{U}}$ indicating whether $\mathcal{U}$ accepted a protocol run, which is only locally used.
$\mathfrak{I}$	Set of $\mathsf{pk}_{\mathcal{U}}$s which already have a balance token issued
$\mathfrak{E}$	Set of $\mathsf{pk}_{\mathcal{U}}$s which have been eliminated due to not accepting a protocol run
$\mathfrak{R}$	Set of tuples $(\mathsf{pk}_{\mathcal{U}}, qid)$ where $\mathsf{pk}_{\mathcal{U}}$ is already registered for qid
$\mathfrak{S}_{\mathcal{U}}$	Set of bulletin tokens $\tau_{\mathcal{V}}$ corresponding to reports submitted by $\mathcal{U}$
$\mathfrak{Q}_{\mathcal{U}}$	Set of $(qid, \tau_{\mathcal{SP}}^{\mathcal{U},qid})$ for $\mathcal{U}$
$\mathfrak{Q}_{qid}$	Set of all report counter tokens $\tau_{\mathcal{SP}}$ that have been observed by the experiment during `RegisterUser` for qid, independent of $\mathcal{U}$
$corrupt_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{U}})$	Stores the output of last Corrupt (used within report limit mechanism and modified in HonSubmitReport)

$corrupt_{\mathcal{I}}(\mathsf{pk}_{\mathcal{U}})$	Same for Stores the output of last Corrupt (used within the incentive mechanism and is reset after a subsequent MalCollect or MalRedeem call)
$\mathfrak{S}'_{\mathcal{U}}$	Set of bulletin tokens $\tau'_{\mathcal{V}}$, where the token contains a simulated commitment and the simulation randomness instead of a real commitment and opening value
$\mathfrak{S}^*_{\mathcal{U}}$	Set of bulletin tokens $\tau_{\mathcal{V}}$, where the token contain a real commitment and opening value (generated after a corruption of $\mathsf{pk}_{\mathcal{U}}$, where $\mathsf{pk}_{\mathcal{U}}$ was already registered for the qid the report has been submitted for)

In the experiments, the following oracles are available to $\mathcal{A}$. A formal Definition is given in Figures 7.13 to 7.18.

HonUser

This oracle allows $\mathcal{A}$ to create a new user identity for an honest user played by the challenger. $\mathcal{A}$ obtains $\mathsf{pk}_{\mathcal{U}}$.

RealHonIssue/SimHonIssue

With this oracle, $\mathcal{A}$ can play the `Issue` protocol with a user identified by $\mathsf{pk}_{\mathcal{U}}$. The oracle checks whether $\mathsf{pk}_{\mathcal{U}}$ already has been issued a balance token or was eliminated due to rejecting a previous protocol run before invoking the `Issue` protocol. The oracle also initializes the user's balance $w_{\mathcal{U}}$ with 0. In SimHonIssue, the user simulator gets $\mathsf{pk}_{\mathcal{U}}$ as `Issue` is identifying, but not $\mathsf{sk}_{\mathcal{U}}$.

RealHonRegisterUser/SimHonRegisterUser

With this oracle $\mathcal{A}$ can have an honest user identified by $\mathsf{pk}_{\mathcal{U}}$ registering for a new query identity qid. The oracle validates that $\mathsf{pk}_{\mathcal{U}}$ has not registered for qid before and has not been eliminated. Moreover, $\mathcal{A}$ is allowed to choose the reporting limit max_{qid} for qid. Note that as the RA is played by the adversary, different report limits max_{qid} may be used for the same qid. In SimHonRegisterUser the user simulator gets $\mathsf{pk}_{\mathcal{U}}$, as `RegisterUser` is identifying, but not $\mathsf{sk}_{\mathcal{U}}$. Additionally, it has to store the counter value $ctr^{\mathcal{U}}_{qid}$ of the user as the actual report counter token is not stored for this user. However, the oracle maintains a list $\mathfrak{Q}_{qid}$ with all report counter tokens issued for qid.

RealHonSubmitReport/SimHonSubmitReport

Using this oracle, $\mathcal{A}$ can have an honest user identified by $\mathsf{pk}_{\mathcal{U}}$ invoke the `SubmitReport` protocol for a qid and m chosen by $\mathcal{A}$. The oracle validates that $\mathsf{pk}_{\mathcal{U}}$ is registered for qid, has not been eliminated and that the report limit has not been exceeded before invoking the protocol. In RealHonSubmitReport, after a successful run, $\tau^{\mathcal{U},qid}_{\mathcal{SP}}$ is updated and the bulletin token $\tau_{\mathcal{V}}$ is remembered for this user. However, in SimHonSubmitReport, the situation is more complicated. Before the protocol run, the oracle has to check if $\mathcal{U}$ has been corrupted before and if this is the case, if $\mathsf{pk}_{\mathcal{U}}$ was already registered for qid when the corruption occurred and no other successful call to SimHonSubmitReport occurred after the corruption (this is done by removing $(qid, \tau^{\mathcal{U},qid}_{\mathcal{SP}})$

from $\mathfrak{Q}_{\mathcal{U}}$ stored in $corrupt_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{U}})$ after a successful call following a corruption). If this was not the case, then the user simulator is used to invoke `SubmitReport`, which does not get $\mathsf{pk}_{\mathcal{U}}$, $\mathsf{sk}_{\mathcal{U}}$ or $\mathcal{U}$'s $\tau_{\mathcal{SP}}$ but gets a report counter token $\tau^{*}_{\mathcal{SP}}$ selected randomly from all previously observed tokens instead. However, if $\mathcal{U}$ has been corrupted and because of this, the report counter token $\tau_{\mathcal{SP}}$ which will be used in this transaction is already known to $\mathcal{A}$, the oracle runs the real user code with the information that has been returned by Corrupt. In both cases, the oracle has to store the bulletin tokens, however, we have to store the tokens for the simulated and non-simulated case separately as in the simulated case, the bulletin token is independent of the user's identity which has to be respected for subsequent corruptions of $\mathsf{pk}_{\mathcal{U}}$. Additionally, in both cases, the oracle has to update the counter value $ctr^{\mathcal{U}}_{qid}$ for the initial verification to work.

RealHonCollect/SimHonCollect

With this oracle, $\mathcal{A}$ can have the honest user identified by $\mathsf{pk}_{\mathcal{U}}$ collect the incentives for a public bulletin commitment $com_{\mathcal{V}}$. The oracle verifies that $com_{\mathcal{V}}$ has indeed been created during an interaction with $\mathsf{pk}_{\mathcal{U}}$ before and that an incentive token for $\mathsf{pk}_{\mathcal{U}}$ has already been issued and $\mathsf{pk}_{\mathcal{U}}$ has not been eliminated. In RealHonCollect, the real user code is executed and $\tau^{\mathcal{U}}_{\mathcal{I}}$ and $w_{\mathcal{U}}$ are updated accordingly. In SimHonCollect, if there was at least one previous interaction with the ISP impersonated by $\mathcal{A}$ before (no RealHonIssue, RealHonCollect, RealHonRedeem call), the interaction is simulated. In this case, the user simulator only obtains $com_{\mathcal{V}}$ and the message authentication key $\mathsf{k}_{\mathrm{mac}}$ contained in $\tau_{\mathcal{V}}$ but not $\mathsf{pk}_{\mathcal{U}}$ or $\mathsf{sk}_{\mathcal{U}}$ or $\tau_{\mathcal{V}}$ itself. In case the interaction directly follows a corruption of the $\mathsf{pk}_{\mathcal{U}}$, the real user code is executed with the information returned during this corruption. In both cases, the oracle has to update the user's balance $w_{\mathcal{U}}$ accordingly.

RealHonRedeem/SimHonRedeem

This oracle allows $\mathcal{A}$ to have the honest user identified by $\mathsf{pk}_{\mathcal{U}}$ redeem an amount v from his incentive balance $w_{\mathcal{U}}$. The oracle verifies that $\mathsf{pk}_{\mathcal{U}}$ already has been issued a balance token, v does not exceed the corresponding balance and $\mathsf{pk}_{\mathcal{U}}$ has not been eliminated. Then, in RealHonRedeem the user invokes the `Redeem` protocol and $\tau^{\mathcal{U}}_{\mathcal{I}}$ and $w_{\mathcal{U}}$ are updated accordingly. In SimHonRedeem, we again distinguish based on whether there was no interaction between the ISP and $\mathcal{U}$ after the last Corrupt call for $\mathsf{pk}_{\mathcal{U}}$. If there was, then the oracle uses the user simulator to invoke `Redeem` which gets the user's current balance $w_{\mathcal{U}}$ but not $\mathsf{pk}_{\mathcal{U}}$ or $\mathsf{sk}_{\mathcal{U}}$. If the call was directly preceded by a corruption of $\mathsf{pk}_{\mathcal{U}}$, then the oracle executes the real user code for `Redeem` with the information that has been returned by Corrupt.

RealCorrupt/SimCorrupt

$\mathcal{A}$ can use this oracle to corrupt the user identified by $\mathsf{pk}_{\mathcal{U}}$, thus obtaining all its private information. In RealCorrupt, this information is directly returned but in SimCorrupt, the balance tokens $\tau^{\mathcal{U}}_{\mathcal{I}}$, $\tau^{\mathcal{U},qid}_{\mathcal{SP}}$ and the bulletin tokens $\tau_{\mathcal{V}}$ for $\mathsf{pk}_{\mathcal{U}}$ have to be simulated to coincide with the interactions $\mathcal{U}$ is supposed to have previously made with $\mathcal{A}$.

HonUser()

$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$
return $\mathsf{pk}_{\mathcal{U}}$

RealHonIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{I}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ **then**
  $((\tau_{\mathcal{I}}^{\mathcal{U}}, b_{\mathcal{U}}), b_{\mathcal{I}}) \leftarrow \mathtt{Issue}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $\mathfrak{I} \leftarrow \mathfrak{I} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
    $w_{\mathcal{U}} := 0$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

RealHonRegisterUser($\mathsf{pk}_{\mathcal{U}}, qid, max_{qid}$)

if $\{(\mathsf{pk}_{\mathcal{U}}, qid)\} \notin \mathfrak{R}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ **then**
  if $max_{qid} > 0$ **then**
    $((\tau_{\mathcal{SP}}^{\mathcal{U},qid}, b_{\mathcal{U}}), (b_{\mathcal{RA}})) \leftarrow \mathtt{RegisterUser}'\left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, max_{qid}), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \end{matrix}\right\rangle$
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $\mathfrak{R} := \mathfrak{R} \cup \{(\mathsf{pk}_{\mathcal{U}}, qid)\}$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

RealHonSubmitReport($\mathsf{pk}_{\mathcal{U}}, qid, m$)

if $(\mathsf{pk}_{\mathcal{U}}, qid) \in \mathfrak{R}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ and $\mathtt{currentCTR}(\tau_{\mathcal{SP}}^{\mathcal{U},qid}) > 0$ **then**
  $\begin{pmatrix} (\tau_{\mathcal{V}}, \tau_{\mathcal{SP}}^{*}, b_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}}) \end{pmatrix} \leftarrow \mathtt{SubmitReport}\left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}^{\mathcal{U},qid}, m), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix}\right\rangle$
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $\tau_{\mathcal{SP}}^{\mathcal{U},qid} := \tau_{\mathcal{SP}}^{*}$
    $\mathfrak{S}_{\mathcal{U}} := \mathfrak{S}_{\mathcal{U}} \cup \{\tau_{\mathcal{V}}\}$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

Figure 7.13: Transaction unlinkability oracles, part 1. In RealHonRegisterUser, $\mathtt{RegisterUser}'$ is equivalent to $\mathtt{RegisterUser}$ apart from $\mathcal{U}$ additionally verifying that the additional input max_{qid} is consistent with the report limit send by $\mathcal{A}$ during the protocol

RealHonCollect($\mathsf{pk}_{\mathcal{U}}, com_{\mathcal{V}}$)

$\tau_{\mathcal{V}} := \texttt{select}(\mathfrak{S}_{\mathcal{U}}, com_{\mathcal{V}})$
if $\mathsf{pk}_{\mathcal{U}} \in \mathfrak{I}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ and $\tau_{\mathcal{V}} \neq \bot$ **then**
  $((\tau_I^*, b_{\mathcal{U}}, w^*), (t_{\text{mac}}, \text{dstag}, \text{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Collect} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_I^{\mathcal{U}}, w_{\mathcal{U}}, \tau_{\mathcal{V}}), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v) \end{matrix} \right\rangle$
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $\tau_I^{\mathcal{U}} := \tau_I^*$
    $w_{\mathcal{U}} := w^*$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

RealHonRedeem($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathsf{pk}_{\mathcal{U}} \in \mathfrak{I}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ and $v \leq w_{\mathcal{U}}$ **then**
  $((\tau_I', b_{\mathcal{U}}, w'), (\text{dstag}, \text{hid}, b_{\mathcal{I}})) \leftarrow \texttt{Redeem} \left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_I^{\mathcal{U}}, w_{\mathcal{U}}, v), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v) \right\rangle$
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $\tau_I^{\mathcal{U}} := \tau_I'$
    $w_{\mathcal{U}} := w'$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif

RealCorrupt($\mathsf{pk}_{\mathcal{U}}$)

$\mathfrak{Q}_{\mathcal{U}} := \emptyset$
for $(\mathsf{pk}_{\mathcal{U}}^*, qid) \in \mathfrak{R}$ where $\mathsf{pk}_{\mathcal{U}}^* \stackrel{?}{=} \mathsf{pk}_{\mathcal{U}}$ **do**
  $\mathfrak{Q}_{\mathcal{U}} := \mathfrak{Q}_{\mathcal{U}} \cup \{(qid, \tau_{\mathcal{SP}}^{\mathcal{U},qid})\}$
endfor
return $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau_I^{\mathcal{U}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}})$

Figure 7.14: Transaction unlinkability oracles, part 2. Hereby, the `select` algorithm used within RealHonCollect returns the first bulletin token $\tau_{\mathcal{V}}$ from the set of bulletin tokens specified in the first parameter which contains the bulletin commitment $com_{\mathcal{V}}$ specified in the second parameter. If no such element exists, the algorithm returns $\bot$

SimHonIssue($\mathsf{pk}_{\mathcal{U}}$)

if $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{I}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ **then**
  $((\tau_I^{\mathcal{U}}, b_{\mathcal{U}}), b_I) \leftarrow \mathtt{Issue} \left\langle \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_I, \mathsf{pk}_{\mathcal{U}}), \mathcal{A}(\mathsf{pk}_I, \mathsf{sk}_I, \mathsf{pk}_{\mathcal{U}}) \right\rangle$
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $\mathfrak{I} \leftarrow \mathfrak{I} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
    $w_{\mathcal{U}} := 0$
    $corrupt_I := \bot$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

SimHonRegisterUser($\mathsf{pk}_{\mathcal{U}}, qid, max_{qid}$)

if $\{(\mathsf{pk}_{\mathcal{U}}, qid)\} \notin \mathfrak{R}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ **then**
  if $max_{qid} > 0$ **then**
    $\begin{pmatrix} (\tau_{\mathcal{SP}}, b_{\mathcal{U}}), \\ (b_{\mathcal{RA}}, transcript_{\mathcal{RA}}) \end{pmatrix} \leftarrow \mathtt{RegisterUser'} \left\langle \begin{matrix} \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \end{matrix} \right\rangle$
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $\mathfrak{R} := \mathfrak{R} \cup \{(\mathsf{pk}_{\mathcal{U}}, qid)\}$
    $ctr_{qid}^{\mathcal{U}} := max_{qid}$
    $\mathfrak{Q}_{qid} := \mathfrak{Q}_{qid} \cup \{\tau_{\mathcal{SP}}\}$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

Figure 7.15: Transaction unlinkability oracles, part 3. In SimHonRegisterUser, `RegisterUser'` is equivalent to `RegisterUser` apart from $\mathcal{U}_{\mathsf{Sim}}$ additionally verifying that the additional input max_{qid} is consistent with the report limit send by $\mathcal{A}$ during the protocol

SimHonSubmitReport($\mathsf{pk}_{\mathcal{U}}, qid, m$)

if $(\mathsf{pk}_{\mathcal{U}}, qid) \in \mathfrak{R}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ and $ctr^{\mathcal{U}}_{qid} > 0$ **then**
  if $corrupt_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{U}}) \stackrel{?}{=} \bot$ or $((\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}}) := corrupt_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{U}}); \mathtt{select}'(\mathfrak{Q}_{\mathcal{U}}, qid) \stackrel{?}{=} \bot)$ **then**
    $\tau^{*}_{\mathcal{SP}} \stackrel{\$}{\leftarrow} \mathfrak{Q}_{qid}$
    $\begin{pmatrix} (\tau'_{\mathcal{V}}, \tau^{**}_{\mathcal{SP}} b_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}}) \end{pmatrix} \leftarrow \mathtt{SubmitReport} \left\langle \begin{matrix} \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, qid, \tau^{*}_{\mathcal{SP}}, m), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix} \right\rangle$
    if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
      $\mathfrak{S}'_{\mathcal{U}} := \mathfrak{S}'_{\mathcal{U}} \cup \{\tau'_{\mathcal{V}}\}$
    endif
  else
    $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}}) := corrupt_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{U}})$
    $\tau^{\mathcal{U},qid}_{\mathcal{SP}} := \mathtt{select}'(\mathfrak{Q}_{\mathcal{U}}, qid)$
    $\begin{pmatrix} (\tau_{\mathcal{V}}, \tau^{*}_{\mathcal{SP}}, b_{\mathcal{U}}), \\ (c, com_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}}) \end{pmatrix} \leftarrow \mathtt{SubmitReport} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau^{\mathcal{U},qid}_{\mathcal{SP}}, m), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \end{matrix} \right\rangle$
    if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
      $\mathfrak{S}^{*}_{\mathcal{U}} := \mathfrak{S}^{*}_{\mathcal{U}} \cup \{\tau_{\mathcal{V}}\}$
      $\mathfrak{Q}'_{\mathcal{U}} := \mathfrak{Q}_{\mathcal{U}} \backslash \{qid, \tau^{\mathcal{U},qid}_{\mathcal{SP}}\}$
      $corrupt_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{U}}) := (\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}'_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}})$
    endif
  endif
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $ctr^{\mathcal{U}}_{qid} := ctr^{\mathcal{U}}_{qid} - 1$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

Figure 7.16: Transaction unlinkability oracles, part 4. Hereby, $\mathtt{select}'(\mathfrak{Q}_{\mathcal{U}}, qid)$ returns $\tau^{\mathcal{U},qid}_{\mathcal{SP}}$ where $(qid, \tau^{\mathcal{U},qid}_{\mathcal{SP}}) \in \mathfrak{Q}_{\mathcal{U}}$ or $\bot$ if no such tuple exists.

SimHonCollect($\mathsf{pk}_{\mathcal{U}}, com_{\mathcal{V}}$)

$\tau'_{\mathcal{V}} := \mathtt{select}(\mathfrak{S}'_{\mathcal{U}} \cup \mathfrak{S}_{\mathcal{U}}, com_{\mathcal{V}})$
if $\mathsf{pk}_{\mathcal{U}} \in \mathfrak{I}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ and $\tau'_{\mathcal{V}} \neq \bot$ **then**
  if $corrupt_{\mathcal{I}} \stackrel{?}{=} \bot$ **then**
    $\mathsf{k}_{\mathrm{mac}} := \mathtt{getMacKey}(\tau'_{\mathcal{V}})$
    $((\tau^*_{\mathcal{I}}, b_{\mathcal{U}}, w^*), (t_{\mathrm{mac}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})) \leftarrow \mathtt{Collect} \left\langle \begin{matrix} \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, com_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}}), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v) \end{matrix} \right\rangle$
    if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
      $w_{\mathcal{U}} := w^*$
    endif
  else
    $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}}) := corrupt_{\mathcal{I}}(\mathsf{pk}_{\mathcal{U}})$
    $\tau_{\mathcal{V}} := \mathtt{select}(\mathfrak{S}_{\mathcal{U}}, com_{\mathcal{V}})$
    $((\tau^*_{\mathcal{I}}, b_{\mathcal{U}}, w^*), (t_{\mathrm{mac}}, \mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})) \leftarrow \mathtt{Collect} \left\langle \begin{matrix} \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, w_{\mathcal{U}}, \tau_{\mathcal{V}}), \\ \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v) \end{matrix} \right\rangle$
    if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
      $w_{\mathcal{U}} := w_{\mathcal{U}} + w^*$
      $corrupt_{\mathcal{I}} := \bot$
    endif
  endif
  if $b_{\mathcal{U}} \stackrel{?}{=} 0$ **then**
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif
endif

Figure 7.17: Transaction unlinkability oracles, part 5. Hereby, the `select` algorithm used within RealHonCollect returns the first bulletin token $\tau_{\mathcal{V}}$ from the set of bulletin tokens specified in the first parameter which contains the bulletin commitment $com_{\mathcal{V}}$ specified in the second parameter. If no such element exists, the algorithm returns $\bot$. Additionally, the `getMacKey` algorithm returns the message authentication key $\mathsf{k}_{\mathrm{mac}}$ contained in a bulletin token $\tau_{\mathcal{V}}$. $\tau^*_{\mathcal{I}}$ remains unused.

SimHonRedeem($\mathsf{pk}_{\mathcal{U}}, v$)

if $\mathsf{pk}_{\mathcal{U}} \in \mathfrak{I}$ and $\mathsf{pk}_{\mathcal{U}} \notin \mathfrak{E}$ and $v \leq w_{\mathcal{U}}$ **then**
  if $corrupt_{\mathcal{I}} \stackrel{?}{=} \bot$ **then**
    $((\tau^*_{\mathcal{I}}, b_{\mathcal{U}}, w'), (\mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})) \leftarrow \mathtt{Redeem} \left\langle \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, w_{\mathcal{U}}, v), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v) \right\rangle$
  else
    $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}}) := corrupt_{\mathcal{I}}(\mathsf{pk}_{\mathcal{U}})$
    $((\tau^*_{\mathcal{I}}, b_{\mathcal{U}}, w'), (\mathsf{dstag}, \mathsf{hid}, b_{\mathcal{I}})) \leftarrow \mathtt{Redeem} \left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, w_{\mathcal{U}}, v), \mathcal{A}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v) \right\rangle$
  endif
  if $b_{\mathcal{U}} \stackrel{?}{=} 1$ **then**
    $w_{\mathcal{U}} := w_{\mathcal{U}} - v$
    $corrupt_{\mathcal{I}} := \bot$
  else
    $\mathfrak{E} := \mathfrak{E} \cup \{\mathsf{pk}_{\mathcal{U}}\}$
  endif

SimCorrupt($\mathsf{pk}_{\mathcal{U}}$)

$(\tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}}) \leftarrow \mathsf{Sim}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \mathfrak{R}, \mathfrak{S}'_{\mathcal{U}}, \mathfrak{S}^*_{\mathcal{U}})$
$corrupt_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{U}}) := (\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}})$
$corrupt_{\mathcal{I}}(\mathsf{pk}_{\mathcal{U}}) := (\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}})$
return $(\mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}})$

Figure 7.18: Transaction unlinkability oracles, part 6. In the SimHonCollect oracle, $\tau^*_{\mathcal{I}}$ remains unused.

$\mathrm{Exp}^{\text{TU-real}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{state}_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\text{HonUser,RealHonIssue,RealHonRegisterUser,RealHonSubmitReport,RealHonCollect,RealHonRedeem,RealCorrupt}}(state_0)$

$\mathrm{Exp}^{\text{TU-ideal}}_{\mathrm{GI},\mathcal{A}}(\mathrm{n})$

$(\mathsf{CRS}, \mathsf{td}_{\mathsf{Sim}}) \leftarrow \mathtt{SimSetup}(1^n)$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{state}_0) \leftarrow \mathcal{A}_0(\mathsf{CRS})$
$b \leftarrow \mathcal{A}_1^{\text{HonUser,SimHonIssue,SimHonRegisterUser,SimHonSubmitReport,SimHonCollect,SimHonRedeem,SimCorrupt}}(state_0)$

Figure 7.19: Real and ideal world *transaction unlinkability* experiments. The oracles available to $\mathcal{A}$ within the experiments are defined in Figures 7.13 to 7.18

Definition 7.10 (Transaction unlinkability) *Let* $\mathrm{Exp}^{\text{TU-real}}_{\mathrm{GI},\mathcal{A}}$ *and* $\mathrm{Exp}^{\text{TU-ideal}}_{\mathrm{GI},\mathcal{A}}$ *be defined as in Figure 7.19. An instantiation* GI *of I3PS ensures* transaction unlinkability *if for all adversaries* $\mathcal{A} := (\mathcal{A}_0, \mathcal{A}_1)$*, there exists a negligible function* $\mathsf{negl}(n)$ *such that*

$$\left|\Pr\left[\mathrm{Exp}^{\text{TU-real}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] - \Pr\left[\mathrm{Exp}^{\text{TU-ideal}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1\right]\right| \leq \mathsf{negl}(n)$$

7.4.10 False accusation protection

This property demands that an honest user cannot be falsely accused of double-spending by an adversary that may collude with the RA, SP, and ISP.

In the security experiment, an honest user is created and the same user oracles as specified for the real world experiment of *transaction unlinkability* above (Figures 7.13 and 7.14) are available to the adversary $\mathcal{A}$ to interact with the user. Note that $\mathcal{A}$ does not have access to the HonUser and RealCorrupt oracles. Afterward, the adversary has to output a valid proof of guilt for this user.

$\mathrm{Exp}^{\text{FACP}}_{\mathrm{GI},\mathcal{A}}(n)$

$(\mathsf{CRS}, \mathsf{td}) \leftarrow \mathtt{Setup}(1^n)$
$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \leftarrow \mathtt{SetupRA}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{state}_0) \leftarrow \mathcal{A}_0(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) \leftarrow \mathtt{UGen}(\mathsf{CRS})$
$\Pi \leftarrow \mathcal{A}_1^{\text{RealHonIssue,RealHonRegisterUser,RealHonSubmitReport,RealHonCollect,RealHonRedeem}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, state_0)$
return $(\mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \Pi)$ or $\mathtt{VerifyGuilt}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \Pi))$

Figure 7.20: *False accusation protection* experiment for I3PS. The oracles available to $\mathcal{A}$ are defined in Figure 7.13 and Figure 7.14. They can only be called for $\mathsf{pk}_{\mathcal{U}}$, as $\mathcal{A}$ does not have access to the HonUser oracle to create new user identities.

Definition 7.11 (False accusation protection) *Let* $\mathrm{Exp}^{\text{FACP}}_{\mathrm{GI},\mathcal{A}}$ *be defined as in Figure 7.20. An instantiation* GI *of our model ensures* false accusation protection *if for all* PPT *adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr\left[\mathrm{Exp}^{\text{FACP}}_{\mathrm{GI},\mathcal{A}}(n) \stackrel{?}{=} 1\right] \leq \mathsf{negl}(n)$$

7.5 Instantiation

In this chapter, we give an instantiation for the I3PS model. We make use of PEPSICo in a white box manner and two adapted versions of BBA+, one for the incentive mechanism similar to the interim model and one for the report limitation mechanism.

In the following, we first specify the building blocks used within our instantiation, then discuss how some of the core mechanisms are designed and lastly, give an instantiation for all the algorithms and protocols defined by the model.

7.5.1 Building blocks

MAC

Within the bulletin mechanism, we a mac to achieve *verifiable delivery*. Therefore, we make use of a EUF-CMA *secure* MAC $(\mathtt{Gen}, \mathtt{Mac}, \mathtt{Verify})$ denoted by M.

Group setup

Let `SetupGrp` generate the description gp of a bilinear group for which the SXDH problem is assumed to be hard and the q-DDHI assumption holds over G_1.

$$gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathtt{SetupGrp}(1^n)$$

All the other building blocks used by the incentive mechanism and report limitation mechanism make use of `SetupGrp` as their common group setup algorithm.

NIZK

We make use of $F_{gp}^{(1)}$-, $F_{gp}^{(2)}$-, $F_{gp}^{(3)}$-, $F_{gp}^{(4)}$-, and $F_{gp}^{(5)}$-extractable NIZK proof systems, denoted by $P1$, $P2$, $P3$, $P4$ and $P5$, respectively. $F_{gp}^{(1)}$, $F_{gp}^{(2)}$, $F_{gp}^{(3)}$, $F_{gp}^{(4)}$, and $F_{gp}^{(5)}$ behave as the identity function with respect to group elements and map elements from $\mathbb{Z}_p$ either to G_1 or G_2 depending on whether these are used as exponents of a G_1 or G_2 element within the language. The proof systems have a shared setup algorithm $\mathrm{CRS}_{\mathsf{pok}} \leftarrow \mathtt{SetupPoK}(gp)$ but we make use of two different CRS. One shared between $P1$, $P2$ and $P3$, the proof systems used within the incentive mechanism and a separate one for $P4$ and $P5$ which are used within the report limitation mechanism ($\mathrm{CRS}_{\mathsf{pok}_{\mathcal{I}}}$ and $\mathrm{CRS}_{\mathsf{pok}_{\mathcal{SP}}}$, respectively). We make use of an SXDH-based instantiation of Groth-Sahai proofs [GS08].

Homomorphic commitments

We use three equivocable, additively homomorphic commitment schemes $C_{\mathcal{I}}$, $C_{\mathcal{SP}}$ and $C_{\mathcal{V}}$.

$C_{\mathcal{I}}$ is used within the incentive mechanism. The message space of the scheme is $\mathbb{Z}_p^4$ and the commitment space G_2. Decommitment values are from G_1. Furthermore, the commitment is $F_{gp}^{\mathcal{I}}$-binding, where $F_{gp}^{\mathcal{I}}$ is a function mapping $m := (m_0, m_1, m_2, m_3)$ to $M := (g_1^{m_0}, g_1^{m_1}, g_1^{m_2}, g_1^{m_3})$ and, thus, G_1^4 is the implicit message space.

$C_{\mathcal{SP}}$ is used within the report limitation mechanism but has the same properties as $C_{\mathcal{I}}$. We call its binding function $F_{gp}^{\mathcal{SP}}$, but it is equivalent to $F_{gp}^{\mathcal{I}}$.

$C_{\mathcal{V}}$ is used within the bulletin mechanism. Its message space is $\mathbb{Z}_p$ and the commitment space G_2. Decommitment values are from G_1. It is $F_{gp}^{\mathcal{V}}$-binding, where $F_{gp}^{\mathcal{V}}$ is a function mapping m to $M := g_1^m$. Thus, it implicit message space is G_1.

The verification equations have to be compatible with the proof system, as users will have to prove that they can open commitments to specific values. Therefore, the commitment scheme from Abe et al.

[Abe+11] is used. As CAdd and DAdd coincide with the multiplication of commitments and decommitment values, respectively, these operations are denoted by '$\cdot$'.

Signatures

We use signatures on commitments to attest the validity of the corresponding balance and report counter tokens. Moreover, users required to prove their knowledge of a valid signature without revealing it. We, therefore, use two EUF-CMA secure signature schemes $S_{\mathcal{I}}$ and $S_{\mathcal{SP}}$ for messages in G_2 and $G_1 \times G_2$, respectively, which have to be compatible with the proof systems. As a concrete instantiation, the structure-preserving signature scheme from [Abe+11] is used. In this scheme, the signatures are in $G_2^2 \times G_1$.

PKE

We use a IND-CPA secure PKE scheme E_τ to generate the hidden user ID hid. Its message space is G_1 and it has to be compatible with our proof systems. We use the ElGamal encryption scheme [ElG84].

PEPSICo

We extend the generic PEPSICo instantiation given in [GMP14]. It is based upon an IBE scheme $E_{\text{PI}} := (\texttt{Setup}, \texttt{Extract}, \texttt{Enc}, \texttt{Dec})$ and a PRF $\{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$. We replace this PRF with a VRF $F = (\texttt{SetupGrp}, \texttt{SetupVRF}, \texttt{Gen}, \texttt{Eval}, \texttt{Prove}, \texttt{Verify})$ in the same bilinear group setting as our other building blocks where the underlying PRF is of the form $\mathbb{Z}_p \times \mathbb{Z}_p \to G_1$. This restricts the query identity space to $\mathbb{Z}_p$, but has the advantage that the VRF is compatible with the other building blocks we use.

The complete construction of the extended scheme PI is given in Figure 7.21. We include a SetupGrp and SetupVRF algorithm, equal to their counterparts in F to generate the CRS for the VRF. In addition, we compute the VRF key pair $(\mathsf{pk}_F, \mathsf{sk}_F)$ in PI.Setup and include include them in the keys of the RA. We further extend PI.RegisterMN to output the proof of correctness π_{qid} together with the VRF image regMN_{qid} of *qid*. To be able to verify that regMN_{qid} has been computed honestly by the RA we include a Verify algorithm that executes F.Verify. Note that the *node privacy*, *query privacy* and *transaction unlinkability* of PI still hold as a VRF is pseudorandom. However, in the *node privacy* experiment, CorruptMN outputs the proof π_{qid} in addition to regMN_{qid}.

For the IBE scheme, we need to choose the message length large enough for the report data and a MAC key to be contained within a single message, eg. $\{0,1\}^{2n}$. Note that this can also be achieved by using hybrid encryption, that is, using the IBE scheme to encrypt the key of a symmetric encryption scheme and encrypting the message symmetrically under this key. A possible instantiation of the IBE scheme would be the IND-CCA *secure* variant of Boneh and Franklin [BF01]. For the VRF, we use the construction from [Bel+09] which is based on the q-DDHI assumption and Groth-Sahai proofs.

$\mathsf{SetupGrp}(1^n)$

$gp \leftarrow F.\mathsf{SetupGrp}(1^n)$
return gp

$\mathsf{SetupVRF}(gp)$

$\mathsf{CRS} \leftarrow F.\mathsf{SetupVRF}(gp)$
return CRS

$\mathsf{Setup}(\mathsf{CRS})$

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow F_{\mathrm{PI}}.\mathsf{Setup}(1^n)$
$(\mathsf{pk}_F, \mathsf{sk}_F) \leftarrow F.\mathsf{Gen}(\mathsf{CRS})$
$\mathsf{pk}_{\mathcal{RA}} := (\mathsf{mpk}, \mathsf{pk}_F)$
$\mathsf{sk}_{\mathcal{RA}} := (\mathsf{msk}, \mathsf{sk}_F)$
return $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}})$

$\mathsf{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

$(T_{qid}, \pi_{qid}) \leftarrow F.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{pok}_{\mathrm{PI}}}, \mathsf{sk}_F, qid)$
$\mathsf{regMN}_{qid} := T_{qid}$
return $(\mathsf{regMN}_{qid}, \pi_{qid})$

$\mathsf{Verify}(\mathsf{pk}_{\mathcal{RA}}, qid, \mathsf{regMN}_{qid}, \pi)$

$b \leftarrow F.\mathsf{Verify}(\mathsf{pk}_F, qid, \mathsf{regMN})$
return b

$\mathsf{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$

$\mathsf{sk}_{id} \leftarrow E_{\mathrm{PI}}.\mathsf{Extract}(\mathsf{mpk}, \mathsf{msk}, qid)$
$T_{qid} \leftarrow F.\mathsf{Eval}(\mathsf{CRS}_{\mathsf{pok}_{\mathrm{PI}}}, \mathsf{sk}_F, qid)$
$\mathsf{regQ}_{qid} := (\mathsf{sk}_{id}, T_{qid})$
return regQ_{qid}

$\mathsf{ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, st_{qid})$

$(T, c_1) := c$
if $T \stackrel{?}{=} st_{qid}$ **then**
  return c
else return $\perp$
endif

$\mathsf{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$

$c_1 := E_{\mathrm{PI}}.\mathsf{Enc}(\mathsf{mpk}, qid, m)$
$c := (T_{qid}, c_1)$
return c

$\mathsf{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid)$

$st_{qid} := T_{qid}$
return st_{qid}

$\mathsf{DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, c)$

$(T, c_1) := c$
$m \leftarrow E_{\mathrm{PI}}.\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{qid}, c_1)$
return m

Figure 7.21: Generic instantiation of PEPSICo extended with a VRF. In ReportData, *qid* has to be interpreted as a bitstring

7.5.2 Used mechanisms

Incentive mechanism

As in the interim model, we use BBA+ as our incentive mechanism. During the `Issue` protocol, users are issued a balance token of the form $\tau_I = (com, d, \sigma, s, u_1)$, Hereby, $com \leftarrow C_I.\mathtt{Com}(\mathsf{CRS}_I, (s, w, \mathsf{sk}_\mathcal{U}, u_1))$ is a commitment to the token's version number s, he token's balance w, the user's secret key $\mathsf{sk}_\mathcal{U}$ and the user randomness u_1. Moreover, the token contains the commitment value d required to open the commitment and a signature $\sigma \leftarrow S_I.\mathtt{Sign}(\mathsf{sk}_I, com)$ which is generated by the ISP and attests the validity of the token. With each `Collect` and `Redeem` protocol run, the user obtains a new token containing the modified balance.

Using the zero-knowledge proof systems $P1$, $P2$ and $P3$, a user can prove to the *ISP* that the commitment *com* is of the correct for without revealing secret information. Moreover, due to the homomorphism of the commitments, the ISP can modify the balance contained within the commitment without knowing its content. The user, however, can validate that the commitment has been modified correctly.

Report limitation mechanism

For the report limitation mechanism, we use a modified version of BBA+. Instead of an incentive balance w, the commitment within a report counter token contains a counter ctr. Initially, ctr is the maximum number of data reports max_{qid} allowed to be submitted for qid. This number is decreased with every submitted report and is required to be larger than 0 to submit another report. The modified token is of the form $\tau_{\mathcal{SP}} = (com, d, \sigma, s, u_1, \mathsf{regMN}_{qid}, ctr)$ where $com \leftarrow C_{\mathcal{SP}}.\mathtt{Com}(\mathsf{CRS}_{\mathsf{com}_\mathcal{V}}, (s, ctr, \mathsf{sk}_\mathcal{U}, u_1))$. To bind the report counter token to qid, the corresponding mobile node registration value regMN_{qid} is signed together with the commitment, ie. $\sigma \leftarrow S_{\mathcal{SP}}.\mathtt{Sign}(\mathsf{sk}_{\mathcal{SP}}, \mathsf{regMN}_{qid}, com)$. The signature key pair $(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$, which is required to generate and verify σ, is shared by the RA and the SP as both parties need to be able to sign commitments.

Double-spending detection mechanism

The same double-spending detection mechanism is used in both, the incentive and the report counter mechanism. Each token is associated with a specific token version number s which is revealed during `SubmitReport`, `Collect` and `Redeem` protocol runs. Moreover, the user has to reveal a value t which is of the form $t = \mathsf{sk}_\mathcal{U} u_2 + u_1$, whereby $\mathsf{sk}_\mathcal{U}$ is the user's secret key, u_2 is a random value supplied by the SP or ISP and u_1 is the user randomness contained within the commitment. As long as each token is only used once, t looks completely random to the SP or ISP, however, if double-spending is committed, that is, an old token is used within a transaction, the same token version number $s_0 = s_1$ and user randomness u_1 has to be used during the protocol run, but the random value from the SP or ISP differs with overwhelming probability ($u_2 \neq u_2'$). Therefore, two different values $t = \mathsf{sk}_\mathcal{U} u_2 + u_1$ and $t' = \mathsf{sk}_\mathcal{U} u_2' + u_1$ are obtained. As u_2 and u_2' are known, $\mathsf{sk}_\mathcal{U}$ can be extracted which proves that the user identified by $\mathsf{pk}_\mathcal{U}$ has committed double-spending.

Trapdoor-linkability

Within each SubmitReport, Collect and Redeem protocol run, a hidden user ID *hid* is revealed to the SP or ISP. Hereby, $hid = E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, \mathsf{pk}_\mathcal{U})$ is an encryption of the user's public key $\mathsf{pk}_\mathcal{U}$ and is proven to contain the correct public key during the zero-knowledge proof used within the incentive or report limitation mechanism. The public encryption key pk_τ is contained in the CRS and the private encryption key sk_τ is the trapdoor td that allows to link transactions.

Bulletin mechanism

During SubmitReport, the user uses $C_\mathcal{V}$ to compute a commitment $com_\mathcal{V}$ on his secret key $\mathsf{sk}_\mathcal{U}$, which is sent to the SP. During the NIZK proof $P4$ used for the report limitation mechanism, the user additionally proves that $com_\mathcal{V}$ can be opened to the same $\mathsf{pk}_\mathcal{U}$ as contained within the report counter token. The user stores $com_\mathcal{V}$ and the corresponding decommitment value $d_\mathcal{V}$. The SP passes $com_\mathcal{V}$ on to the querier, who is then able to post incentives for $com_\mathcal{V}$. During the NIZK proof $P2$, which is used to collect incentives, the user proves that $com_\mathcal{V}$ can be opened to the same $\mathsf{pk}_\mathcal{U}$ as contained within his balance token. This ensures that incentives can only be collected by the same user that submitted the report.

Delivery verification

To enable the querier to verify that posted incentives have indeed been delivered correctly, during SubmitReport, the user includes a newly generated MAC key k_{mac} in the plaintext for the PEPSICo report. Therefore, only authorized queriers can obtain k_{mac}. The user remembers k_{mac} together with the bulletin commitment $com_\mathcal{V}$. As there might be multiple queriers and, therefore, multiple incentives posted for $com_\mathcal{V}$, an additional value $r_\mathcal{V}$, randomly chosen by the querier, is added to each incentive post. The user computes $t_{\text{mac}} \leftarrow M.\mathsf{Mac}(k_{\text{mac}}, com_\mathcal{V} \| r_\mathcal{V})$ and sends it to the SP. The SP can then use t_{mac} to proof to the querier that the incentives have been submitted successfully. Hereby, the message $com_\mathcal{V} \| r_\mathcal{V}$ is already known to the querier. Hereby, $r_\mathcal{V}$ ensures that the proof cannot be reused by the SP for other incentive posts for the same $com_\mathcal{V}$.

7.5.3 Setup algorithms

Setup

The Setup algorithm given in Figure 7.22 generates the CRS of the system. It has the form $\mathsf{CRS} := (gp, \mathsf{CRS}_{\mathsf{com}_\mathcal{I}}, \mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{CRS}_{\mathsf{com}_\mathcal{V}}, \mathsf{pk}_\tau, \mathsf{CRS}_{\mathsf{pok}_\mathcal{I}}, \mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}})$ where gp are the group parameters defining the bilinear group used in the commitments and Groth-Sahai proofs. $\mathsf{CRS}_{\mathsf{com}_\mathcal{I}}$, $\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}$ and $\mathsf{CRS}_{\mathsf{com}_\mathcal{V}}$ are the CRS's for the commitment schemes used in the incentive mechanism, the report limitation mechanism and the bulletin mechanism, respectively. Furthermore, $\mathsf{CRS}_{\mathsf{pok}_\mathcal{I}}$ and $\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}$ are the CRS's for the proof systems used within the incentive mechanism and report limitation mechanism, respectively. The trapdoor td is the secret decryption key for the encryption scheme used in both, the incentive and the report limitation mechanism.

$\mathsf{Setup}(1^n)$

$gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathsf{SetupGrp}(1^n)$
$\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}} \leftarrow C_{\mathcal{I}}.\mathsf{Gen}(gp)$
$\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}} \leftarrow C_{\mathcal{SP}}.\mathsf{Gen}(gp)$
$\mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}} \leftarrow C_{\mathcal{V}}.\mathsf{Gen}(gp)$
$(\mathsf{sk}_\tau, \mathsf{pk}_\tau) \leftarrow E_\tau.\mathsf{Gen}(gp)$
$\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}} \leftarrow \mathsf{SetupPoK}(gp)$
$\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}} \leftarrow \mathsf{SetupPoK}(gp)$
$\mathsf{CRS}_{\mathsf{pok}_{\mathrm{PI}}} \leftarrow \mathrm{PI}.\mathsf{SetupVRF}(gp)$
$\mathsf{CRS} := (gp, \mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}},$
$\quad \mathsf{pk}_\tau, \mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, \mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, \mathsf{CRS}_{\mathsf{pok}_{\mathrm{PI}}})$
$\mathsf{td} := \mathsf{sk}_\tau$
return $(\mathsf{CRS}, \mathsf{td})$

$\mathsf{SetupRA}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \leftarrow \mathrm{PI}.\mathsf{Setup}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow S.\mathsf{Gen}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) := ((\mathsf{CRS}, \mathsf{pk}_{\mathsf{Sig}}), \mathsf{sk}_{\mathsf{Sig}})$
return $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$

Figure 7.22: Setup and SetupRA algorithm

$\mathsf{IGen}(\mathsf{CRS})$

$(\mathsf{pk}_{\mathsf{Sig}}, \mathsf{sk}_{\mathsf{Sig}}) \leftarrow S.\mathsf{Gen}(\mathsf{CRS})$
$(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}) := ((\mathsf{CRS}, \mathsf{pk}_{\mathsf{Sig}}), \mathsf{sk}_{\mathsf{Sig}})$
return $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$

$\mathsf{UGen}(\mathsf{CRS})$

$y \leftarrow \mathbb{Z}_p$
$(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}) := (g_1^y, y)$
return $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$

Figure 7.23: IGen and UGen algorithm

SetupRA

The SetupRA algorithm generates $(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}})$, the secret and private key pair of the RA as well as $(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$, the key pair it shares with the SP. The first one is used to generate mobile node and querier registration values, the second is the signature key that is used to sign the report counter tokens within the report limitation mechanism. Hereby, $\mathsf{pk}_{\mathcal{SP}}$ contains the CRS.

IGen

The issuer key generation algorithm IGen (Figure 7.23) generates a signature key pair $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$. With $\mathsf{sk}_{\mathcal{I}}$, the ISP signs the commitments within the balance tokens to attest their validity and with $\mathsf{pk}_{\mathcal{I}}$, the signatures can be verified. In addition, $\mathsf{pk}_{\mathcal{I}}$ contains the CRS.

UGen

The user key generation algorithm UGen (Figure 7.23) generates the user's public key $\mathsf{pk}_{\mathcal{U}}$ and private key $\mathsf{sk}_{\mathcal{U}}$ used during the protocols. Hereby, the public key is the G_1 element corresponding to $\mathsf{sk}_{\mathcal{U}}$.

7.5.4 User protocols

Issue

With the balance token issuing protocol Issue (Figure 7.24), a user identified by $\mathsf{pk}_{\mathcal{U}}$ can obtain an initial balance token, which is required to collect or redeem incentive points. We use the Issue protocol from BBA+. First, the user generates a $C_{\mathcal{I}}$ commitment com' to the balance 0 and his private key $\mathsf{sk}_{\mathcal{U}}$, together with some random values s' and u_1. He then sends the commitment to the ISP and proofs that it contains the correct balance and the $\mathsf{sk}_{\mathcal{U}}$ corresponding to the $\mathsf{pk}_{\mathcal{U}}$ which the user used to identify himself towards the ISP. The ISP verifies the proof, and if it is valid, uses the homomorphism of $C_{\mathcal{I}}$ to add a random share to s' to ensure the token version number $s = s' + s''$ is random even if one of the two parties is cheating. It then signs the resulting commitment com. The user computes his balance token $\tau_{\mathcal{I}}$ from the values supplied by the ISP and checks if it is indeed a valid token linked to his user ID and with the balance 0 before he outputs the token.

$\texttt{Issue}\ \langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}) \rangle$

$\mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$		$\mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$
$s', u_1 \leftarrow \mathbb{Z}_p$		
$(com', d') \leftarrow C_{\mathcal{I}}.\texttt{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s', 0, \mathsf{sk}_{\mathcal{U}}, u_1))$		
$stm \leftarrow (com', \mathsf{pk}_{\mathcal{U}})$		
$wit \leftarrow (g_2^{\mathsf{sk}_{\mathcal{U}}}, g_1^{s'}, g_1^{u_1}, d')$		
$\pi \leftarrow P1.\texttt{Prove}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, wit)$	$\xrightarrow{com', \pi}$	
		$stm \leftarrow (com', \mathsf{pk}_{\mathcal{U}})$
		if $P1.\texttt{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return 0
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{I}}.Com(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s'', 0, 0, 0))$
		$com := com' \cdot com''$
	$\xleftarrow{com, d'', \sigma, s''}$	$\sigma \leftarrow S.\texttt{Sign}(\mathsf{sk}_{\mathcal{I}}, com)$
$s := s' + s'' \mod p$		
$d := d' \cdot d''$		
$\tau_{\mathcal{I}} := (com, d, \sigma, s, u_1)$		
if $\texttt{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, 0) \stackrel{?}{=} 0$ **then**		
return $(\perp, 0)$		
endif		
return $(\tau_{\mathcal{I}}, 1)$		**return** 1

Figure 7.24: Issue protocol. The language for $P1$ is defined in Figure 7.25 and the $\texttt{UVer}_{\mathcal{I}}$ algorithm in Figure 7.26

$$L^{(1)}_{\mathsf{pk}_{\mathcal{I}}} := \left\{ (com', \mathsf{pk}_{\mathcal{U}}) \;\middle|\; \begin{array}{l} \exists SKU \in G_2; \\ S', U_1, d' \in G_1 : \\ C.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, com', d', (S', 1, \mathsf{pk}_{\mathcal{U}}, U_1) \stackrel{?}{=} 1 \\ e(\mathsf{pk}_{\mathcal{U}}, g_1) \stackrel{?}{=} e(g_1, SKU) \end{array} \right\}$$

Figure 7.25: Language for $P1$ (used within `Issue`)

$\mathsf{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w)$

$(com, d, \sigma, s, u_1) := \tau_{\mathcal{I}}$

if $\mathsf{pk}_{\mathcal{U}} \stackrel{?}{=} g_1^{\mathsf{sk}_{\mathcal{U}}} \wedge C_{\mathcal{I}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, com, d, (g_1^s, g_1^w, \mathsf{pk}_{\mathcal{U}}, g_1^{u_1})) \stackrel{?}{=} 1 \wedge S.\mathsf{Verify}(\mathsf{pk}_{\mathcal{I}}, com, \sigma) \stackrel{?}{=} 1$ **then**
  return 1
else
  return 0
endif

Figure 7.26: $\mathsf{UVer}_{\mathcal{I}}$ algorithm

RegisterUser

The `RegisterUser` protocol (Figure 7.27) allows a user to obtain all the report counter token $\tau_{\mathcal{SP}}$ necessary elements to submit reports for a query identity qid. The RA controls who can register for a query identity and should ensure that no user can register multiple times for the same query identity.

The protocol combines the mobile node registration from PEPSICo and a modified version of the issue protocol from BBA+. In addition to the protocol, the RA must make sure that no user can register multiple times for the same query identity. In the protocol, the RA sends the report limit max_{qid}, the maximum number of reports that users are allowed to submit for this query identity, to the user. The user generates a commitment com' on max_{qid}, his secret key $\mathsf{sk}_{\mathcal{U}}$ and some additional randomness required for the double-spending detection. He proofs in zero-knowledge to the registration authority that the commitment contains the correct values (cf. Figure 7.28). The RA adds a random share to the token version number s' contained in com', resulting in a new commitment com, and computes the mobile node registration value regMN_{qid} for qid as well as a proof π_{qid} that regMN_{qid} has been computed correctly. It then signs the com together with regMN_{qid} and sends regMN_{qid}, the signature σ and the added randomness s'' to the user. The user then computes the corresponding report counter token $\tau_{\mathcal{SP}}$ and verifies that it is valid and that regMN_{qid} was computed correctly, before outputting $\tau_{\mathcal{SP}}$.

SubmitReport

The `SubmitReport` protocol (Figure 7.31) adapts the `Accum` protocol of BBA+ to enforce the report limit. The user generates the report c containing the report data m and a new message authentication key

$\texttt{RegisterUser}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid})\right\rangle$

$\mathcal{U}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid)$		$\mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid})$
	$\xleftarrow{max_{qid}}$	
$s', u_1 \leftarrow \mathbb{Z}_p$		
$(com', d') \leftarrow C_{\mathcal{SP}}.\texttt{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, (s', max_{qid}, \mathsf{sk}_{\mathcal{U}}, u_1))$		
$stm \leftarrow (com', \mathsf{pk}_{\mathcal{U}}, g_1^{max_{qid}})$		
$wit \leftarrow (g_2^{\mathsf{sk}_{\mathcal{U}}}, g_1^{s'}, g_1^{u_1}, d')$		
$\pi \leftarrow P4.\texttt{Prove}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, stm, wit)$	$\xrightarrow{com', \pi}$	
		$stm \leftarrow (com', \mathsf{pk}_{\mathcal{U}}, g_1^{max_{qid}})$
		if $P4.\texttt{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return 0
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{SP}}.Com(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, (s'', 0, 0, 0, 0))$
		$com := com' \cdot com''$
		$(\mathsf{regMN}_{qid}, \pi_{qid}) \leftarrow \mathrm{PI}.\texttt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
	$\xleftarrow{\mathsf{regMN}_{qid}, com, d'', \sigma, s'', \pi_{qid}}$	$\sigma \leftarrow S.\texttt{Sign}(\mathsf{sk}_{\mathcal{SP}}, \mathsf{regMN}_{qid}, com)$
$s := s' + s'' \mod p$		
$d := d' \cdot d''$		
$\tau_{\mathcal{SP}} := (com, d, \sigma, s, u_1, \mathsf{regMN}_{qid}, max_{qid})$		
if $\texttt{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{SP}}) \stackrel{?}{=} 0$		
$\vee\ \mathrm{PI}.\texttt{Verify}(\mathsf{pk}_{\mathcal{RA}}, qid, \mathsf{regMN}_{qid}, \pi_{qid}) \stackrel{?}{=} 0$ **then**		
return $(\perp, \perp, 0)$		
endif		
return $(\tau_{\mathcal{SP}}, 1)$		**return** 1

Figure 7.27: `RegisterUser` protocol. The $\texttt{UVer}_{\mathcal{SP}}$ algorithm used within the protocol is given in Figure 7.29 and the language for $P4$ in Figure 7.28

$$L^{(4)}_{\mathsf{pk}_{\mathcal{SP}}} := \left\{ (com', \mathsf{pk}_{\mathcal{U}}, MAX_{qid}) \;\middle|\; \begin{array}{l} \exists SKU \in G_2; \\ S', U_1, d' \in G_1 : \\ C.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, com', d', (S', MAX_{qid}, \mathsf{pk}_{\mathcal{U}}, U_1) \stackrel{?}{=} 1 \\ e(\mathsf{pk}_{\mathcal{U}}, g_1) \stackrel{?}{=} e(g_1, SKU) \end{array} \right\}$$

Figure 7.28: Languages for $P4$ (used within `RegisterUser`)

$\mathsf{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{SP}})$

$(com, d, \sigma, s, u_1, \mathsf{regMN}_{qid}, ctr) := \tau_{\mathcal{SP}}$

if $\mathsf{pk}_{\mathcal{U}} \stackrel{?}{=} g_1^{\mathsf{sk}_{\mathcal{U}}} \wedge C_{\mathcal{SP}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, com, d, (g_1^s, g_1^{ctr}, \mathsf{pk}_{\mathcal{U}}, g_1^{u_1})) \stackrel{?}{=} 1$

$\wedge\, S.\mathsf{Verify}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{regMN}_{qid}, com, \sigma) \stackrel{?}{=} 1$ **then**

return 1

else

return 0

endif

Figure 7.29: $\mathsf{UVer}_{\mathcal{SP}}$ algorithm

$$L^{(5)}_{\mathsf{pk}_{\mathcal{SP}}} := \left\{ \begin{pmatrix} com', com_{\mathcal{V}}, S, t, \\ u_2, \mathsf{regMN}_{qid}, hid \end{pmatrix} \;\middle|\; \begin{array}{l} \exists com \in G_2; \\ \sigma \in G_2^2 \times G_1; \\ \mathsf{pk}_{\mathcal{U}}, U_1, d, S', U_1', d', d_{\mathcal{V}}, CTR \in G_1; \\ \mathsf{sk}_{\mathcal{U}}, u_1, r \in \mathbb{Z}_p : \\ E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, \mathsf{pk}_{\mathcal{U}}; r) \stackrel{?}{=} hid \\ C_{\mathcal{SP}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, com, d, (S, CTR, \mathsf{pk}_{\mathcal{U}}, U_1)) \stackrel{?}{=} 1 \\ C_{\mathcal{SP}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, com', d', (S', CTR, \mathsf{pk}_{\mathcal{U}}, U_1')) \stackrel{?}{=} 1 \\ C_{\mathcal{V}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}}, com_{\mathcal{V}}, d_{\mathcal{V}}, \mathsf{pk}_{\mathcal{U}}) \stackrel{?}{=} 1 \\ S.\mathsf{Verify}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{regMN}_{qid}, com, \sigma) \stackrel{?}{=} 1 \\ CTR \neq 1 \\ \mathsf{pk}_{\mathcal{U}} \stackrel{?}{=} g_1^{sk_{\mathcal{U}}} \\ U_1 \stackrel{?}{=} g_1^{u_1} \\ t \stackrel{?}{=} \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \mod p \end{array} \right\}$$

Figure 7.30: Language for P5 (used within `SubmitReport`)

$\texttt{SubmitReport}\ \langle \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}, m), \mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \rangle$

$\mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid, \tau_{\mathcal{SP}}, m)$		$\mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$
	$\xleftarrow{u_2}$	$u_2 \leftarrow \mathbb{Z}_p$
$\mathsf{k}_{\mathrm{mac}} \leftarrow M.\mathsf{Gen}(1^n)$		
$(com, d, \sigma, s, u_1, \mathsf{regMN}_{qid}, ctr) := \tau_{\mathcal{SP}}$		
$c \leftarrow \mathrm{PI}.\mathsf{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m \Vert \mathsf{k}_{\mathrm{mac}})$		
$(com_{\mathcal{V}}, d_{\mathcal{V}}) \leftarrow C_{\mathcal{V}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}}, \mathsf{sk}_{\mathcal{U}})$		
$t := \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \mod p$		
$r, s', u_1' \leftarrow \mathbb{Z}_p$		
$\mathsf{hid} := E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, \mathsf{pk}_{\mathcal{U}}; r)$		
$(com', d') \leftarrow C_{\mathcal{SP}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, (s', ctr, \mathsf{sk}_{\mathcal{U}}, u_1'))$		
$stm := (com', com_{\mathcal{V}}, g_1^s, t, u_2, \mathsf{regMN}_{qid}, \mathsf{hid})$		
$wit := (com, \sigma, \mathsf{pk}_{\mathcal{U}}, g_1^{u_1}, d, g_1^{s'}, g_1^{u_1'}, d', d_{\mathcal{V}}, g_1^{ctr}, \mathsf{sk}_{\mathcal{U}}, u_1, r)$		
$\pi \leftarrow P5.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, stm, wit)$	$\xrightarrow{c, com_{\mathcal{V}}, com', s, t, \pi, \mathsf{hid}}$	
		$z := (t, u_2)$
		$\mathsf{dstag} := (s, z)$
		$(\mathsf{regMN}_{qid}, c_1) := c$
		$stm := (com', g_1^s, t, u_2, \mathsf{regMN}_{qid}, \mathsf{hid})$
		if $P5.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return $(\perp, \perp, \perp, 0)$
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{SP}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, (s'', -1, 0, 0, 0))$
		$com^* := com' \cdot com''$
	$\xleftarrow{com^*, d'', \sigma^*, s''}$	$\sigma^* \leftarrow S.\mathsf{Sign}(\mathsf{sk}_{\mathcal{SP}}, \mathsf{regMN}_{qid}, com^*)$
$s^* := s' + s'' \mod p$		
$d^* := d' \cdot d''$		
$ctr^* := ctr - 1$		
$u_1^* := u_1'$		
$\tau_{\mathcal{SP}}^* := (com^*, d^*, \sigma^*, s^*, u_1^*, \mathsf{regMN}_{qid}, ctr^*)$		
if $\mathsf{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{SP}}^*) \stackrel{?}{=} 0$ **then**		
return $(\perp, 0)$		
endif		
$\tau_{\mathcal{V}} := (com_{\mathcal{V}}, d_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}})$		
return $(\tau_{\mathcal{V}}, \tau_{\mathcal{SP}}^*, 1)$		**return** $(c, \mathrm{com}_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, 1)$

Figure 7.31: SubmitReport protocol. The $\mathsf{UVer}_{\mathcal{SP}}$ algorithm used within the protocol is given in Figure 7.29

k_{mac} using the `ReportData` algorithm form PEPSICo and computes a commitment $com_{\mathcal{V}}$ to his public key, which is required for the bulletin mechanism. The rest of the protocol largely corresponds to the BBA+ `Accum`. The user re-randomizes the commitment within the report counter token and sends it together with c, $com_{\mathcal{V}}$ and some additional information required to provide trapdoor-linkability and double-spending detection to the SP. Moreover, he proves in zero-knowledge that the new commitment com' contains the same query identity and report counter ctr as the original commitment for which the user has a valid signature from the SP or RA. In addition, it is proven that the bulletin commitment $com_{\mathcal{V}}$ can be opened to the same $\mathsf{pk}_{\mathcal{U}}$ as com' and com and that ctr not 0 (by verifying that $g_1^{ctr} \neq 1$), meaning that the user did not exceed his report limit. If the proof is accepted, the SP creates and signs a new commitment with the report counter reduced by one (using the homomorphism of the commitment). The new commitment and signature, together with its randomness, is sent to the user which now can compute a new report counter token. Furthermore, the user outputs a bulletin token $\tau_{\mathcal{V}}$ consisting of the commitment $com_{\mathcal{V}}$, the corresponding decommitment value $d_{\mathcal{V}}$ and the message authentication key k_{mac}. The SP outputs the data report c, the bulletin commitment $\mathsf{com}_{\mathcal{V}}$, the double-spending tag dstag and an encryption hid of the user's public key.

Collect

The `Collect` protocol (Figure 7.32) allows a user to collect incentives that have been posted for a report previously submitted by this user. It combines the `Accum` protocol of BBA+, to add the incentive value to the balance of the user's token, with the bulletin collection mechanism.

First, the ISP sends a random u_2 together with the $r_{\mathcal{V}}$ from the querier and the incentive value v that will be added to the user's balance. The user computes t required for the double-spending detection mechanism, hid required for trapdoor-linkability and a new commitment com' containing the same values as the commitment com contained in his balance token apart from a new token version number s' and user randomness u_1'. Within the NIZK proof, he proves that com' contains the correct values, t and hid are generated correctly and he knows a signature σ on com. The user additionally proves that the bulletin commitment $com_{\mathcal{V}}$ can be opened to the same user secret key as included within com, to verify that he submitted the corresponding report (cf. Figure 7.33). This ensures that the incentives cannot be collected by a different user. The ISP verifies the proof and adds a random share s'' to the token version number of com' and the incentive value v to its balance. He signs the resulting commitment com^* and sends it together with the signature, s'' and d'', which is required to compute the decommitment value d^* of com^*, to the user. Upon receiving those values, the user computes a new balance token τ_I^* and verifies that it contains the new balance $w + v$, where w was the old incentive balance of the user. Moreover, at the end of the protocol, the user computes a MAC of $com_{\mathcal{V}}$ and $r_{\mathcal{V}}$, attesting that the incentives have been correctly delivered and send it to the ISP.

$\texttt{Collect}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, \tau_{\mathcal{V}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v)\right\rangle$

$\mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, \tau_{\mathcal{V}})$		$\mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, com_{\mathcal{V}}, r_{\mathcal{V}}, v)$
	$\xleftarrow{u_2, r_{\mathcal{V}}, v}$	$u_2 \leftarrow \mathbb{Z}_p$
$(com, d, \sigma, s, u_1) := \tau_{\mathcal{I}}$		
$t := \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \mod p$		
$r, s', u_1' \leftarrow \mathbb{Z}_p$		
$\mathsf{hid} := E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, \mathsf{pk}_{\mathcal{U}}; r)$		
$(com', d') \leftarrow C_{\mathcal{I}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s', w, \mathsf{sk}_{\mathcal{U}}, u_1'))$		
$(com_{\mathcal{V}}, d_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}}) := \tau_{\mathcal{V}}$		
$stm := (com', com_{\mathcal{V}}, g_1^s, t, u_2, \mathsf{hid})$		
$wit := (com, \sigma, g_1^w, \mathsf{pk}_{\mathcal{U}}, g_1^{u_1}, d, g_1^{s'}, g_1^{u_1'}, d', d_{\mathcal{V}}, \mathsf{sk}_{\mathcal{U}}, u_1, r)$		
$\pi \leftarrow P2.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, wit)$	$\xrightarrow{com', s, t, \pi, \mathsf{hid}}$	
		$z := (t, u_2)$
		$\mathsf{dstag} := (s, z)$
		$stm := (com', com_{\mathcal{V}}, g_1^s, t, u_2, \mathsf{hid})$
		if $P2.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return $(\bot, \bot, \bot, 0)$
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{I}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s'', v, 0, 0))$
		$com^* := com' \cdot com''$
		$\sigma^* \leftarrow S.\mathsf{Sign}(\mathsf{sk}_{\mathcal{I}}, com^*)$
	$\xleftarrow{com^*, d'', \sigma^*, s''}$	
$s^* := s' + s'' \mod p$		
$d^* := d' \cdot d''$		
$w^* = w + v$		
$u_1^* := u_1'$		
$\tau_{\mathcal{I}}^* := (com^*, d^*, \sigma^*, s^*, u_1^*)$		
if $\mathsf{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}^*, w^*) \stackrel{?}{=} 0$ **then**		
return $(\bot, 0)$		
endif		
$t_{\mathrm{mac}} \leftarrow M.\mathsf{Mac}(\mathsf{k}_{\mathrm{mac}}, com_{\mathcal{V}} \| r_{\mathcal{V}})$	$\xrightarrow{t_{\mathrm{mac}}}$	
return $(\tau_{\mathcal{I}}^*, 1, w^*)$		**return** $(t_{\mathrm{mac}}, \mathsf{dstag}, \mathsf{hid}, 1)$

Figure 7.32: `Collect` protocol. The language for $P2$ is defined in Figure 7.33

$$L^{(2)}_{\mathsf{pk}_{\mathcal{I}}} := \left\{ (com', \mathsf{com}_{\mathcal{V}}, S, t, u_2, hid) \;\middle|\; \begin{array}{l} \exists com \in G_2; \\ \sigma \in G_2^2 \times G_1; \\ W, \mathsf{pk}_{\mathcal{U}}, U_1, d, S', U_1', d', d_{\mathcal{V}} \in G_1; \\ \mathsf{sk}_{\mathcal{U}}, u_1, r \in \mathbb{Z}_p : \\ E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, \mathsf{pk}_{\mathcal{U}}; r) \stackrel{?}{=} hid \\ C_{\mathcal{I}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, com, d, (S, W, \mathsf{pk}_{\mathcal{U}}, U_1)) \stackrel{?}{=} 1 \\ C_{\mathcal{I}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, com', d', (S', W, \mathsf{pk}_{\mathcal{U}}, U_1')) \stackrel{?}{=} 1 \\ C_{\mathcal{V}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}}, com_{\mathcal{V}}, d_{\mathcal{V}}, pk_{\mathcal{U}}) \\ S.\mathtt{Verify}(\mathsf{pk}_{\mathcal{I}}, com, \sigma) \stackrel{?}{=} 1 \\ \mathsf{pk}_{\mathcal{U}} \stackrel{?}{=} g_1^{sk_{\mathcal{U}}} \\ U_1 \stackrel{?}{=} g_1^{u_1} \\ t \stackrel{?}{=} \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \mod p \end{array} \right\}$$

Figure 7.33: Language for P2 (used within `Collect`)

Redeem

The `Redeem` protocol (Figure 7.34) is nearly equivalent to the `Verify` Protocol of BBA+. It allows the user to redeem a subset of the incentive points accumulated on his balance token and reveals the balance of the token to the ISP, wherefore it can be verified that the balance exceeds the amount that should be withdrawn. It is very similar to `Collect`, with the exception that the G_1 element of the user's balance W is in the statement of the NIZK proof, there is no bulletin commitment $com_{\mathcal{V}}$ involved and the incentive value v is subtracted from the balance instead of added to it.

7.5.5 Querier protocols

RegisterQ

The querier registration protocol (Figure 7.36) is used by a querier to register for a query identity *qid*. The RA generates the querier registration value regMN_{qid}, which is required to compute the subscription token and to decrypt data reports, and sends it to the querier. The querier then additionally computes the subscription token st_{qid} for *qid*, therefore obtaining all the information required to collect reports for this query identity.

CollectReports

To collect all the submitted reports for a query identity *qid* from the SP, the querier executes the `CollectReports` protocol (Figure 7.37). First, the querier sends the subscription token st_{qid} to the SP.

$\mathsf{Redeem}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, v), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v)\right\rangle$

$\mathcal{U}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}, w, v)$		$\mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v)$
	$\xleftarrow{u_2}$	$u_2 \leftarrow \mathbb{Z}_p$
$(com, d, \sigma, s, u_1) := \tau_{\mathcal{I}}$		
$t := \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \mod p$		
$r, s', u_1' \leftarrow \mathbb{Z}_p$		
$\mathsf{hid} := E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, \mathsf{pk}_{\mathcal{U}}; r)$		
$(com', d') \leftarrow C_{\mathcal{I}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s', w, \mathsf{sk}_{\mathcal{U}}, u_1'))$		
$stm := (com', g_1^s, t, u_2, \mathsf{hid}, g_1^w)$		
$wit := (com, \sigma, \mathsf{pk}_{\mathcal{U}}, g_1^{u_1}, d, g_1^{s'}, g_1^{u_1'}, d',$ $\mathsf{sk}_{\mathcal{U}}, u_1, r)$		
$\pi \leftarrow P3.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, wit)$	$\xrightarrow{com', s, t, \pi, \mathsf{hid}}$	
		$z := (t, u_2)$
		$\mathsf{dstag} := (s, z)$
		$stm := (com', g_1^s, t, u_2, \mathsf{hid}, g_1^w)$
		if $P2.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, \pi) \stackrel{?}{=} 0$ or $w < v$ **then**
		return $(\perp, \perp, 0)$
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{I}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}},$ $(s'', -v, 0, 0))$
		$com^* := com' \cdot com''$
		$\sigma^* \leftarrow S.\mathsf{Sign}(\mathsf{sk}_{\mathcal{I}}, com^*)$
	$\xleftarrow{com^*, d'', \sigma^*, s''}$	
$s^* := s' + s'' \mod p$		
$d^* := d' \cdot d''$		
$w^* = w + v$		
$u_1^* := u_1'$		
$\tau_{\mathcal{I}}^* := (com^*, d^*, \sigma^*, s^*, u_1^*)$		
if $\mathsf{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}^*, w^*) \stackrel{?}{=} 0$ **then**		
return $(\perp, 0)$		
endif		
return $(\tau_{\mathcal{I}}^*, 1, w^*)$		**return** $(\mathsf{dstag}, \mathsf{hid}, 1)$

Figure 7.34: Redeem protocol. The language for $P3$ is defined in Figure 7.35

$$L^{(3)}_{\mathsf{pk}_{\mathcal{I}}} := \left\{ (com', S, t, u_2, hid, W) \;\middle|\; \begin{array}{l} \exists com \in G_2; \\ \sigma \in G_2^2 \times G_1; \\ \mathsf{pk}_{\mathcal{U}}, U_1, d, S', U_1', d' \in G_1; \\ \mathsf{sk}_{\mathcal{U}}, u_1, r \in \mathbb{Z}_p : \\ E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, \mathsf{pk}_{\mathcal{U}}; r) \stackrel{?}{=} hid \\ C_{\mathcal{I}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, com, d, (S, W, \mathsf{pk}_{\mathcal{U}}, U_1)) \stackrel{?}{=} 1 \\ C_{\mathcal{I}}.\mathsf{Open}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, com', d', (S', W, \mathsf{pk}_{\mathcal{U}}, U_1')) \stackrel{?}{=} 1 \\ S.\mathtt{Verify}(\mathsf{pk}_{\mathcal{I}}, com, \sigma) \stackrel{?}{=} 1 \\ \mathsf{pk}_{\mathcal{U}} \stackrel{?}{=} g_1^{sk_{\mathcal{U}}} \\ U_1 \stackrel{?}{=} g_1^{u_1} \\ t \stackrel{?}{=} \mathsf{sk}_{\mathcal{U}} u_2 + u_1 \mod p \end{array} \right\}$$

Figure 7.35: Language for P3 (used within Redeem)

$\mathtt{RegisterQ}\left\langle \mathcal{Q}(\mathsf{pk}_{\mathcal{RA}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}) \right\rangle$

$\mathcal{Q}(\mathsf{pk}_{\mathcal{RA}}, qid)$		$\mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}})$
	$\xrightarrow{qid}$	
	$\xleftarrow{\mathsf{regQ}_{qid}}$	$\mathsf{regQ}_{qid} \leftarrow \mathrm{PI}.\mathtt{RegisterQ}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
$st_{qid} \leftarrow \mathrm{PI}.\mathtt{SubscribeQuery}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid)$		
return $(\mathsf{regQ}_{qid}, st_{qid})$		**return** 1

Figure 7.36: RegisterQ protocol

$\texttt{CollectReports}\left\langle \mathcal{Q}(\mathsf{pk}_{\mathcal{RA}}, qid, \mathsf{regQ}_{qid}, st_{qid}), \mathcal{SP}(\mathsf{pk}_{\mathcal{RA}}, \mathfrak{C}) \right\rangle$

$\mathcal{Q}(\mathsf{CRS}, \mathsf{pk}_{\mathcal{RA}}, qid, \mathsf{regQ}_{qid}, st_{qid})$		$\mathcal{SP}(\mathsf{pk}_{\mathcal{RA}}, \mathfrak{C})$
	$\xrightarrow{st_{qid}}$	
		$\mathfrak{C}^* := \emptyset$
		for $(c, com_{\mathcal{V}}) \in \mathfrak{C}$ **do**
		$c^* \leftarrow \mathrm{PI.ExecuteQuery}(\mathsf{pk}_{\mathcal{RA}}, c, st_{qid})$
		if $c \stackrel{?}{=} c^*$ **then**
		$\mathfrak{C}^* := \mathfrak{C}^* \cup \{(c, com_{\mathcal{V}})\}$
		endif
		endfor
	$\xleftarrow{\mathfrak{C}^*}$	
$\mathfrak{M} := \emptyset$		
for $(c, com_{\mathcal{V}}) \in \mathfrak{C}^*$ **do**		
$m \Vert \mathsf{k}_{\mathrm{mac}} \leftarrow \mathrm{PI.DecodeData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regQ}_{qid}, qid, c)$		
$\mathfrak{M} := \mathfrak{M} \cup \{(m, com_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}})\}$		
endfor		
return $\mathfrak{M}$		**return** 1

Figure 7.37: `CollectReports` protocol. $\mathfrak{C}$ is a list of reports together with their public bulletin information $(c, com_{\mathcal{V}})$

$\texttt{PostIncentives}\left\langle \mathcal{Q}(\mathsf{CRS}, v, \mathsf{com}_{\mathcal{V}}, q), \mathcal{I} \right\rangle$

$\mathcal{Q}(\mathsf{CRS}, v, com_{\mathcal{V}}, q)$		$\mathcal{I}$
$r_{\mathcal{V}} \leftarrow \mathbb{Z}_p$	$\xrightarrow{com_{\mathcal{V}}, v, r_{\mathcal{V}}, q}$	
		if $v \leq 0$ **then**
		return $(\perp, 0, 0, 0)$
		endif
return (h^*)		**return** $(com_{\mathcal{V}}, v, r_{\mathcal{V}}, q)$

Figure 7.38: `PostIncentives` protocol

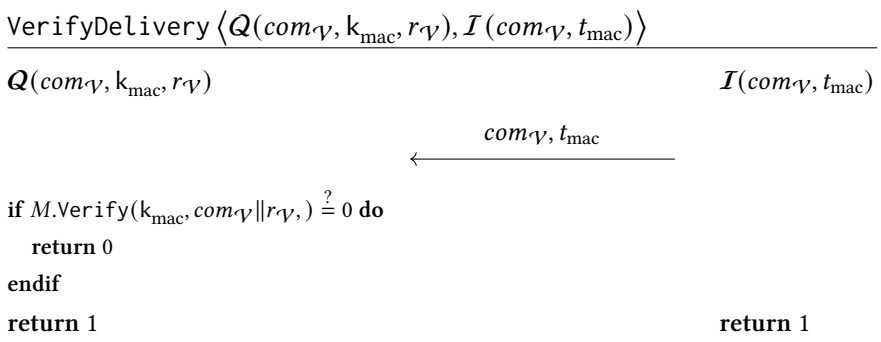

Figure 7.39: `VerifyDelivery` protocol

The SP uses the `ExecuteQuery` algorithm from PEPSICo to determine the reports belonging to the subscription token. These reports, together with their bulletin commitment $\mathsf{com}_{\mathcal{V}}$, are sent to the querier. The querier then uses PI.`DecodeData` to decrypt the reports obtaining m and $\mathsf{k}_{\mathrm{mac}}$. These values are outputted together with $com_{\mathcal{V}}$.

PostIncentives

The `PostIncentives` protocol is used by the querier to put an incentive reward for a received report on the bulletin board, where it can be collected by the corresponding user. Therefore, the querier simply sends the incentive value to be rewarded together with the bulletin commitment $com_{\mathcal{V}}$, identifying the user and a random value $r_{\mathcal{V}}$ required for delivery verification to the ISP. Moreover, the querier sends an indicator q for the quality of the data report, which can help to identify and delete bad data reports from the SP's database. Please note that queriers should not be allowed to reward negative incentive points for obvious reasons. The querier has to remember $r_{\mathcal{V}}$ to verify the delivery of the incentive points later.

VerifyDelivery

With the `VerifyDelivery` protocol, the ISP can convince the querier that the incentive points posted for a report have been delivered. Therefore, he sends the bulletin commitment $com_{\mathcal{V}}$ together with the MAC tag t_{mac}, which was obtained from the user during the execution of the `Collect` protocol, to the querier. The querier checks whether t_{mac} verifies for the message authentication key $\mathsf{k}_{\mathrm{mac}}$ which he obtained during the `CollectReports` protocol.

7.5.6 Double-spending detection algorithms

IdentDS

We use the `IdentDS` algorithm of BBA+ (Figure 7.40), as it works with the double-spending tags from both, the incentives mechanism and the report limitation mechanism. If a token has been used twice, that is, the token version number s included within the double spending tag is identical, then the corresponding user's secret key can be extracted as this ensures that $t = \mathsf{sk}_{\mathcal{U}} u_2 + u_1$ and $t' = \mathsf{sk}_{\mathcal{U}} u_2' + u_1$

$\texttt{IdentDS}(\mathsf{pk}_I, \mathsf{dstag}_0, \mathsf{dstag}_1)$

$(s_1, (t, u_2)) := \mathsf{dstag}_0$
$(s_2, (t', u_2')) := \mathsf{dstag}_1$
if $s_1 \neq s_2$ or $u_2 \stackrel{?}{=} u_2'$ **then**
  return $\perp$
else
  $\mathsf{sk}_{\mathcal{U}} := (t - t') \cdot (u_2 - u_2')^{-1} \mod p$
  $\mathsf{pk}_{\mathcal{U}} := g_1^{\mathsf{sk}_{\mathcal{U}}}$
  $\Pi := \mathsf{sk}_{\mathcal{U}}$
  return $(\mathsf{pk}_{\mathcal{U}}, \Pi)$
endif

$\texttt{VerifyGuilt}(\mathsf{pk}_I, \mathsf{pk}_{\mathcal{U}}, \Pi)$

if $g_1^{\Pi} \stackrel{?}{=} \mathsf{pk}_{\mathcal{U}}$ **then**
  return 1
else
  return 0
endif

Figure 7.40: `IdentDS` and `VerifyGuilt` algorithms

for the same unknown user randomness u_1 and $(t, u_2), (t', u_2')$ contained within the double spending tags. The knowledge of the secret key is then considered a proof for the user's guilt.

VerifyGuilt

The `VerifyGuilt` algorithm (Figure 7.40) is used to verify if Π indeed proofs that the user identified by $\mathsf{pk}_{\mathcal{U}}$ has committed double-spending. In this case, Pi is the secret key $\mathsf{sk}_{\mathcal{U}}$ corresponding to $\mathsf{pk}_{\mathcal{U}}$, that is, $\mathsf{pk}_{\mathcal{U}} = g_1^{\mathsf{sk}_{\mathcal{U}}}$. If this is the case, the algorithm returns 1, else 0.

7.6 Security of the instantiation

In the following, we proof that the instantiation GI of I3PS as defined within Section 7.5 meets the security notions defined in Section 7.4.

7.6.1 Data-hiding

Theorem 7.12 (Data-hiding) *If* PI *is* node-private*, then* GI *is* data-hiding.

Proof We assume there exist a successful adversary $\mathcal{A}$, winning the *data-hiding* experiment $\text{Exp}^{\text{DH}}_{\text{GI},\mathcal{A}}$ (Figure 7.5) with more than negligible advantage. We use this adversary to build an reduction $\mathcal{A}'$ that breaks the *node privacy* property of PEPSICo (Figure 4.3). Our reduction has to be able to simulate the experiment as well as all the oracles available to $\mathcal{A}$ within the experiment.

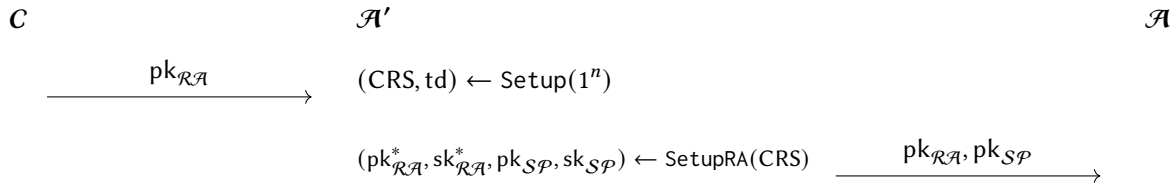

Figure 7.41: Reduction proof for *data-hiding*, part 1

The simulation of $\text{Exp}^{\text{DH}}_{\text{GI},\mathcal{A}}$ looks as follows: First $\mathcal{A}'$ receives the RA's public key $\mathsf{pk}_{\mathcal{RA}}$ from the challenger. It then performs parts of the Setup algorithm and embeds $\mathsf{pk}_{\mathcal{RA}}$. More precisely, $\mathcal{A}'$ generates $(\mathsf{CRS}, \mathsf{td})$ with Setup and $(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$ using SetupRA. Instead of the $\mathcal{RA}$'s key pair $(\mathsf{pk}^*_{\mathcal{RA}}, \mathsf{sk}^*_{\mathcal{RA}})$ generated by the algorithm, the public key $\mathsf{pk}_{\mathcal{RA}}$ from the challenger is used for interactions with $\mathcal{A}$. Therefore, $\mathcal{A}'$ now sends $\mathsf{pk}_{\mathcal{RA}}$ and $\mathsf{pk}_{\mathcal{SP}}$ to $\mathcal{A}$ (Figure 7.41). This is equal to the real experiment from $\mathcal{A}'$'s perspective.

Next, $\mathcal{A}$ can make use of his oracles before outputting two query identity and message pairs (qid_0, m_0) and (qid_1, m_1). Let us first look at the simulation of oracle queries. In the *data-hiding* experiment, the adversary has access to the CorruptSP, MalRegisterUser, MalRegisterQ, SubmitReport and CollectReports oracles. The reduction has access to the CorruptMN, CorruptQ, CorruptSP, ReportData, SubscribeQuery and DecodeData oracles from PEPSICo's *node privacy* experiment. We simulate the oracles available to $\mathcal{A}$ as follows:

CorruptSP

This oracle allows $\mathcal{A}$ to corrupt the secret key material of the SP. Therefore, after calling this oracle $\mathcal{A}$ can impersonate the SP. $\mathcal{A}'$ simulates CorruptSP oracle queries as in Figure 7.42. First, it calls its own CorruptSP oracle, notifying the challenger that the SP has been corrupted. Then it returns the SP's secret key $\mathsf{sk}_{\mathcal{SP}}$ to $\mathcal{A}$. $\mathcal{A}'$ has to remember that the SP has been corrupted to modify its behavior accordingly.

From the view of $\mathcal{A}$, this simulation behaves exactly like CorruptSP in $\text{Exp}^{\text{DH}}_{\text{GI},\mathcal{A}}$.

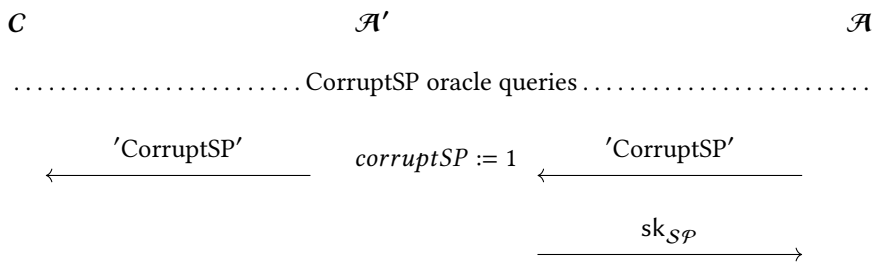

Figure 7.42: Simulation of CorruptSP oracle queries

MalRegisterUser

This oracle allows $\mathcal{A}$ to register a user for a query identity qid with the RA. This user is completely controlled by the adversary, ie. $\mathcal{A}$ obtains the report counter token $\tau_{\mathcal{SP}}$ for qid, which contains the mobile node registration value regMN_{qid}. However, $\mathcal{A}$ is not allowed to query MalRegisterUser for qid_0 or qid_1 which he has to output later in the experiment.

The simulation is shown in Figure 7.43. Given $\mathsf{pk}_{\mathcal{U}}$, qid and max_{qid}, $\mathcal{A}'$ checks whether the user identified by $\mathsf{pk}_{\mathcal{U}}$ has already registered for qid and makes sure that the maximum for report submissions max_{qid} is consistent with previous calls. Then $\mathcal{A}'$ uses its CorruptMN oracle to obtain regMN_{qid} and π_{qid}. $\mathcal{A}'$ is not allowed to use this oracle on qid_0 or qid_1 but the same restrictions apply to $\mathcal{A}$ with respect to MalRegisterUser as discussed above.

Finally, $\mathcal{A}'$ executes the `RegisterUser` protocol from the RA's point of view with $\mathcal{A}$ impersonating the user identified by $\mathsf{pk}_{\mathcal{U}}$. Within the protocol, $\mathcal{A}'$ does not generate the mobile node registration value for qid but instead uses regMN_{qid} and π_{qid} obtained from CorruptMN as described above. Therefore, $\mathcal{A}'$ does not require knowledge of $\mathsf{sk}_{\mathcal{RA}}$ to execute the protocol. For $\mathcal{A}$, this simulation looks identical to the real MalRegisterUser.

MalRegisterQ

This oracle allows $\mathcal{A}$ to register as a querier for the query identity qid. MalRegisterQ can be simulated by $\mathcal{A}'$ through relaying the call to its CorruptQ oracle. For both oracles, the same usage restrictions apply: They are not allowed to be queried for qid_0 or qid_1.

SubmitReport

This oracle allows $\mathcal{A}$ to have an honest user submit a report to the (honest) SP. $\mathcal{A}$ is not allowed to call SubmitReport for qid_0 or qid_1 if CorruptSP is called during the experiment. As it does not return any data, $\mathcal{A}'$ can simulate it by simply remembering the submitted data. Therefore $\mathcal{A}'$ stores any qid, m for which $\mathcal{A}$ has called SubmitReport within the list of submitted reports $\mathfrak{SR}$. Note that as SubmitReport can be called for qid_0 and qid_1 in case the SP is not compromised during the experiment, $\mathcal{A}'$ cannot necessarily generate the corresponding report c.

CollectReports

This oracle allows $\mathcal{A}$ to have an honest querier execute `CollectReports` for a query identity

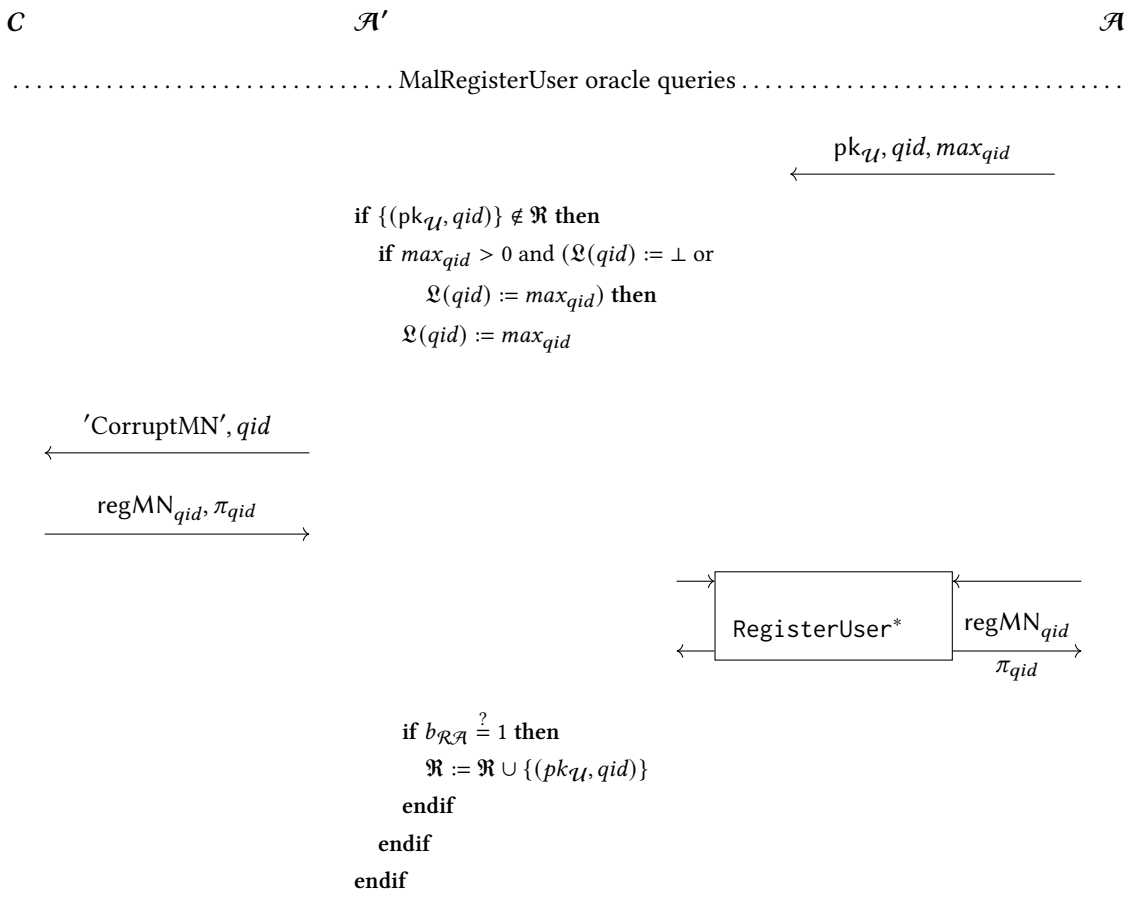

Figure 7.43: Simulation of MalRegisterUser oracle queries. The `RegisterUser`* protocol is identical to `RegisterUser` with the exception that the mobile node registration value regMN_{qid} and the corresponding proof π_{qid}, which are send to $\mathcal{A}$ during the protocol, are not computed but replaced with the values obtained by querying the CorruptMN oracle. The other inputs and outputs are omitted in the figure.

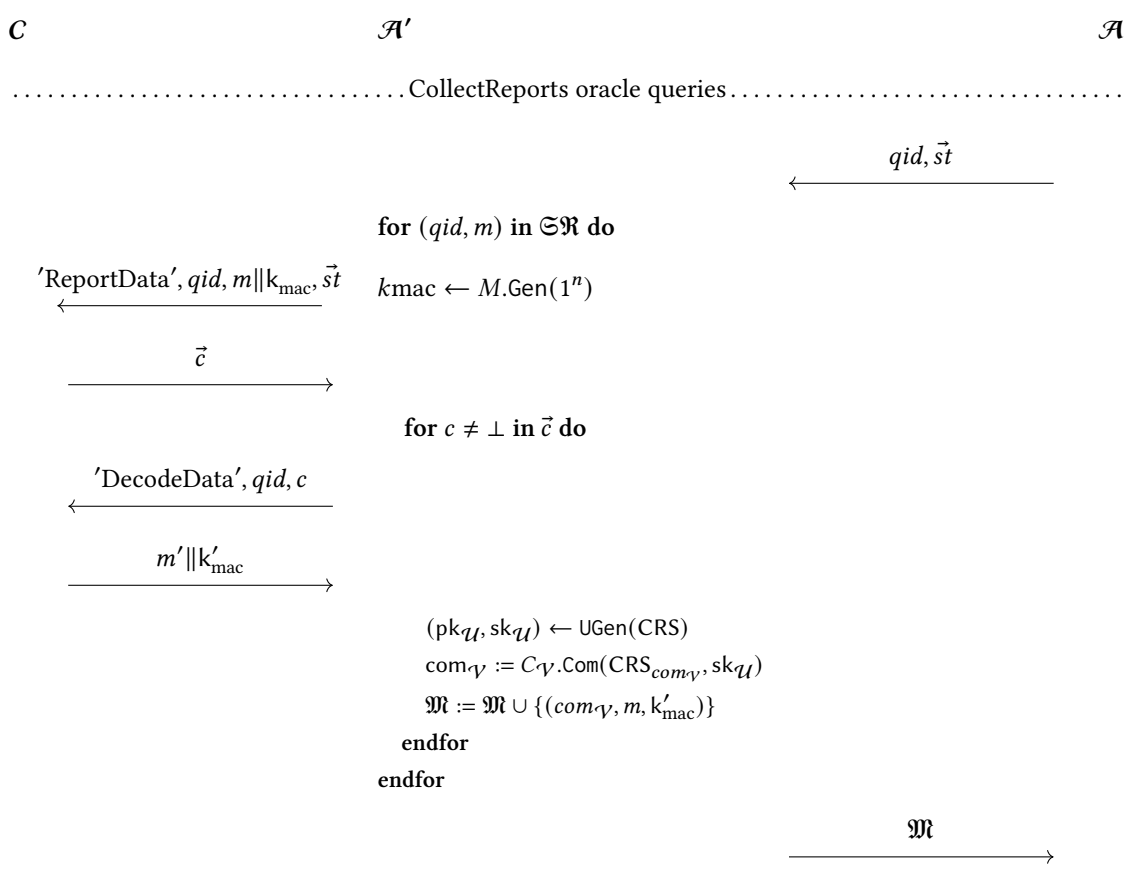

Figure 7.44: Simulation of CollectReports oracle queries for $corruptSP = 0$

a subscription token of his choice and to obtain the resulting output. In combination with SubmitReport, $\mathcal{A}$ can try to forge a subscription token for qid_0 or qid_1 which he could use to win the experiment.

We distinguish between two cases. If the SP has not been corrupted, $\mathcal{A}'$ simulates CollectReports oracle queries a shown in Figure 7.44. Upon obtaining a query identity qid and a vector of subscription tokens $\vec{st}$, $\mathcal{A}'$ calls its ReportData oracle for all the previously submitted reports stored in $\mathfrak{SR}$ together with the subscription token vector $\vec{st}$ and generating a new message authentication key $\mathsf{k}_{\text{mac}}$for each of them. Note that in case the SP is corrupted later within the experiment, ReportData is only called for qid_0 or qid_1 given that $\mathcal{A}$ has called `SubmitReport` for the same query identity and therefore already lost the *data-hiding* experiment. For each element $\neq \perp$ in the report vector $\vec{c}$ returned by ReportData, $\mathcal{A}'$ calls its DecodeData oracle with the query identity qid supplied by the adversary. For each report content m' obtained in this way, $\mathcal{A}'$ generates public bulletin commitment $\mathsf{com}_{\mathcal{V}}$ belonging to a new user identity. A list of all the obtained report contents together with their bulletin commitments and message authentication keys k_{mac} is sent to $\mathcal{A}$. To $\mathcal{A}$, the simulation behaves like the real oracle. He obtains the content of all previously submitted reports for which $\vec{st}$ contains a valid subscription token. Moreover,

each report is accompanied by a bulletin commitment for a newly generated user identity and an independently generated message authentication key.

Let us consider the case that the SP has already been corrupted. In this case, $\mathcal{A}$ should not have queried SubmitReport for qid_0 or qid_1 as else $\mathcal{A}$ would have already lost the experiment. Therefore, $\mathcal{A}'$ can just use its CorruptMN and CorruptQ oracles to obtain the necessary information to execute the real oracle code, except for the invocation of `RegisterUser` within SubmitReport where the mobile node generation value has to be replaced with the value obtained from CorruptMN instead of computed (cf. Simulation of MalRegisterUser).

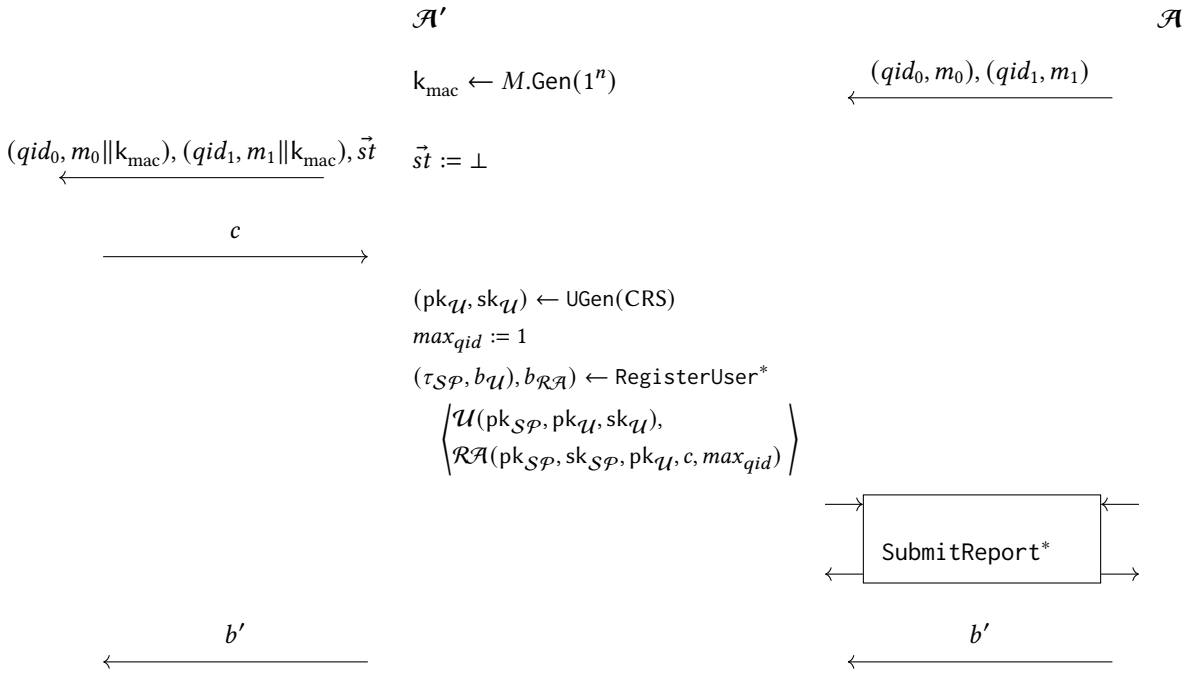

Figure 7.45: Reduction proof for data hiding, part 2 case 1. `RegisterUser`* is a modification of `RegisterUser` where the mobile node registration value is not computed but taken from a report c and the verification of the corresponding proof π_{qid} is ignored. Therefore, the RA's key pair and the query identity is not required to execute the protocol. `SubmitReport`* is a modification of `SubmitReport` where the challenge report c is injected instead of computing a new report

After the adversary outputs (qid_0, m_0) and (qid_1, m_1), the experiment distinguishes whether the SP has already been corrupted and is impersonated by $\mathcal{A}$ or not. In the first case (Figure 7.45), that is the SP has been corrupted, $\mathcal{A}'$ directly forwards the two query identity and message pairs to the challenger together with an empty vector of subscription tokens $\vec{st}$. After obtaining the challenge report $c = (\mathsf{regMN}_{qid}, c_1)$, $\mathcal{A}'$ generates a new user key pair $(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}})$. To generate the report counter token for the simulation of the `SubmitReport` protocol, $\mathcal{A}'$ simulates the execution of `RegisterUser`. Hereby, the regMN_{qid} from the challenge report is used instead of computing the value using PI.`RegisterMN`. Therefore, knowing the corresponding query identity or the RA's secret key is not required.

$\mathcal{A}'$ now simulates the execution of SubmitReport with $\mathcal{A}$, where $\mathcal{A}'$ plays the role of the user identified by $\mathsf{pk}_{\mathcal{U}}$ and $\mathcal{A}$ the role of the corrupted SP. Within the protocol run, $\mathcal{A}'$ does not generate a data report with PI.ReportData but instead uses the challenge report c. As $\tau_{\mathcal{SP}}$ is a valid report counter token for the same query identity as c, there are no complications in generating the zero-knowledge proof in the protocol.

If $\mathcal{A}'$ accepts the protocol run (which is done on the same conditions as for an honest user), it waits for the adversary to output a guess b' for b. This guess is then relayed to the challenger. $\mathcal{A}'$ wins with the same probability then $\mathcal{A}$.

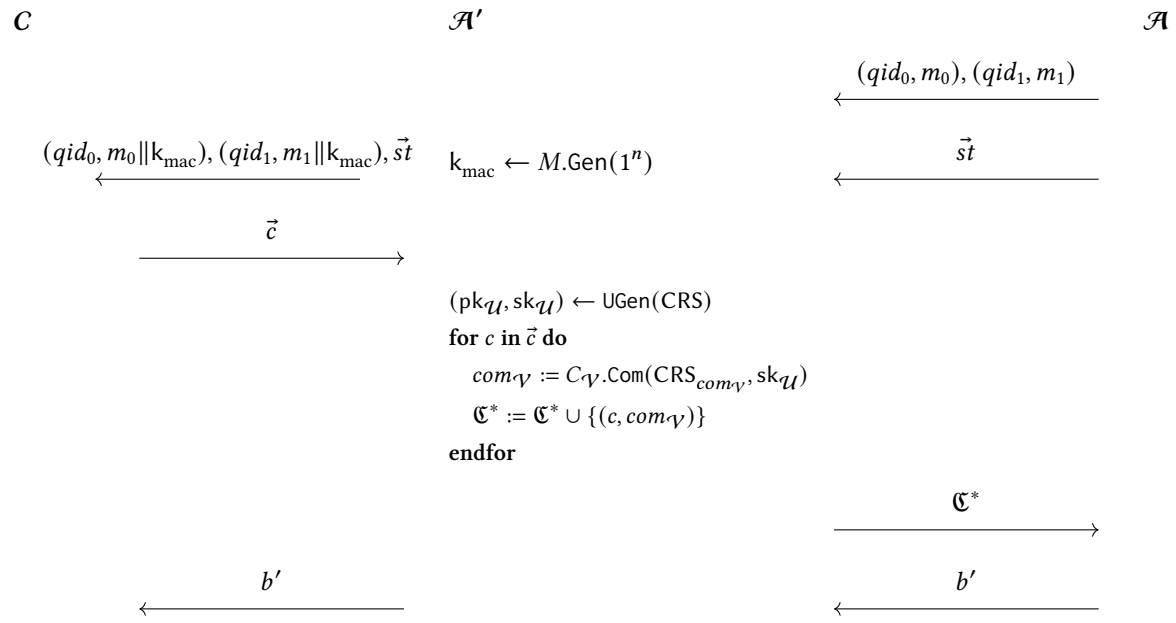

Figure 7.46: Reduction proof for data hiding, part 2 case 2

Now, let us consider the second case (Figure 7.46) where the service provider has not been corrupted when $\mathcal{A}$ outputs (qid_0, m_0) and (qid_1, m_1). In this case, $\mathcal{A}'$ delays forwarding (qid_0, m_0) and (qid_1, m_1) and waits for the adversary to send a vector of subscription tokens $\vec{st}$ as part of the CollectReports protocol where $\mathcal{A}$ plays the role of the querier and $\mathcal{A}'$ the role of the SP. $\mathcal{A}'$ now sends $(qid_0, m_0 \| \mathsf{k}_{\mathrm{mac}})$ and $(qid_1, m_1 \| \mathsf{k}_{\mathrm{mac}})$, for a newly generated message authentication key $\mathsf{k}_{\mathrm{mac}}$, together with $\vec{st}$ to the challenger. After the challenger responded with a vector of reports $\vec{c}$, $\mathcal{A}'$ generates a new user ID. Subsequently, for all reports c in $\vec{c}$, $\mathcal{A}'$ generates a bulletin commitment $com_{\mathcal{V}}$ using this user ID and adds $(c, com_{\mathcal{V}})$ to the list of reports $\mathfrak{C}^*$, which is returned to $\mathcal{A}$. The guess b' that $\mathcal{A}$ outputs for b is relayed to the challenger. Again $\mathcal{A}'$ has the same advantage as $\mathcal{A}$.

Therefore, we have shown that the *node privacy* of PI implies that GI is *data-hiding*, ie. for all $\mathcal{A}, \mathcal{A}'$ there exist negligible functions $\mathsf{negl}, \mathsf{negl}'$ such that

$$\mathsf{Adv}^{\text{NP-CCA}}_{\text{PI},\mathcal{A}'}(n) \leq \mathsf{negl}'(n) \implies \mathsf{Adv}^{\text{DH}}_{\text{GI},\mathcal{A}}(n) \leq \mathsf{negl}(n)$$

7.6.2 Subscription-hiding

Theorem 7.13 (Subscription-hiding) *If* PI *is* query private*, then* GI *is* subscription-hiding.

Proof We reduce the *subscription-hiding* property of our instantiation to the *query privacy* of the underlying PEPSICo instantiation. More precisely, we assume there exists a successful adversary $\mathcal{A}$, winning the *subscription-hiding* experiment of I3PS with more than negligible advantage. We use this adversary to construct an adversary $\mathcal{A}'$ that breaks the *query privacy* property of PEPSICo, which contradicts its security.

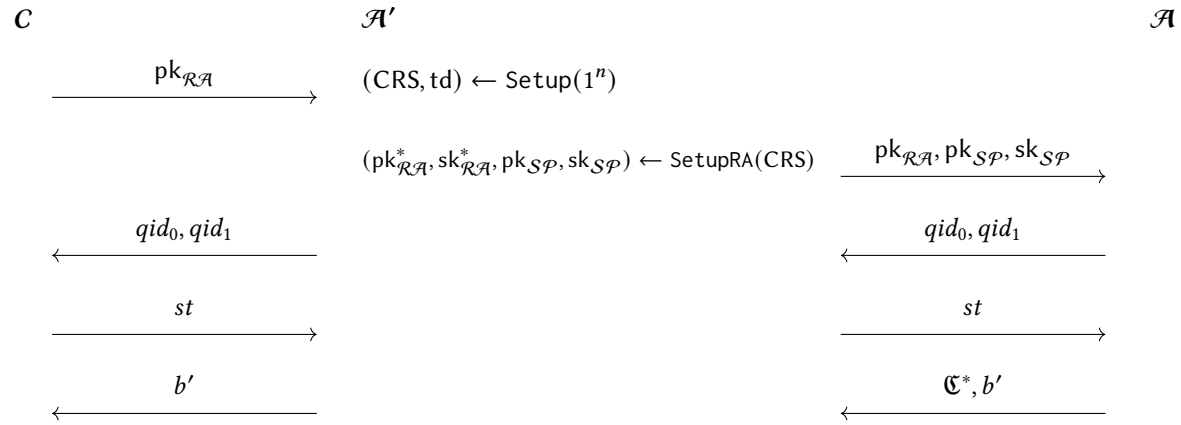

Figure 7.47: Reduction of an adversary on the *subscription-hiding* property of I3PS to the *query privacy* of PEPSICo

The reduction $\mathcal{A}'$ is given in Figure 7.47. It has to generate the CRS and the key pair for the report limitation mechanism and send those together with the public key of the RA, which is obtained from the challenger, to the adversary. After obtaining the query identity pair and relaying it to the challenger, it then executes the `CollectReports` protocol with the adversary but using the challenge subscription token instead of computing one. Moreover, as $\mathcal{A}'$ is not interested in the message list $\mathfrak{M}$, it can abort the protocol after receiving the list of reports $\mathfrak{C}^*$, without encrypting its elements. Lastly, the decision of the adversary is relayed to the challenger.

Furthermore, in the experiment $\mathcal{A}$ has access to the oracles MalRegisterUser and MalRegisterQ, whereas $\mathcal{A}'$ can use the CorruptMN, CorruptQ, ReportData, SubscribeQuery and DecodeData oracles from the *query privacy* experiment from PEPSICo. The oracles available to $\mathcal{A}$ are simulated as in the previous proof (Section 7.6.1). To simulate MalRegisterUser queries, the reduction has to check the conditions, eg. users are only allowed to register once per query identity, and maintain the lists of registered users and previously defined report maximums. It uses its CorruptMN oracle to obtain the mobile node registration value for the queried *qid* and initiates the `RegisterUser` protocol with the adversary. Within this protocol, instead of computing a mobile node registration value, the one obtained from the challenger is used. Therefore, the RA's secret key, which is unknown to the reduction, is not required to execute the protocol.

MalRegisterQ oracle queries can be simulated using the CorruptQ oracle available to the reduction. Keep in mind that $\mathcal{A}$ is not allowed to query MalRegisterUser and MalRegisterQ for qid_0 or qid_1 as well as $\mathcal{A}'$ is not allowed to query CorruptMN and CorruptQ for these two query identities.

Our reduction $\mathcal{A}'$ is successful whenever $\mathcal{A}$ is successful. Therefore, the existence of an adversary $\mathcal{A}$ with more than negligible advantage would contradict the *query privacy* of PEPSICo and, thus, no such adversary exists. More precisely, we have shown that for all PPT adversaries $\mathcal{A}, \mathcal{A}'$

$$\mathsf{Adv}^{\mathrm{QP}}_{\mathrm{PI},\mathcal{A}}(n) \leq \mathsf{negl}(n) \implies \mathsf{Adv}^{\mathrm{QP}}_{\mathrm{GI},\mathcal{A}'}(n) \leq \mathsf{negl}(n)$$

where PI is the PEPSICo instantiation used within the I3PS instantiation GI.

7.6.3 Trapdoor-linkability

Theorem 7.14 (Trapdoor-linkability) *If* GI *and* E_τ *are correct and P2, P3 and P5 are perfectly sound, then* GI *is* trapdoor-linkable.

Proof idea

Completeness

Consider a hidden user ID token hid which the SP outputs after a `SubmitReport` protocol run. This implies that the proof π sent during the protocol verifies, and since $P5$ is sound, hid can therefore be generated using E_τ.Enc. More precisely, there exists $m \in G_1$ and $r \in \mathbb{Z}_p$ such that $E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, m; r) \stackrel{?}{=} \mathsf{hid}$. As the message space for E_τ coincides with the space for user public keys, m is a valid public key $\mathsf{pk}_\mathcal{U}$. By the definition of `UGen`, there exists a corresponding secret key $\mathsf{sk}_\mathcal{U}$.

As we assume GI is correct, an honest user with the key pair $(\mathsf{pk}_\mathcal{U}, \mathsf{sk}_\mathcal{U})$ could have obtained a valid report counter token $\tau_{\mathcal{SP}}$ by executing `SubmitReport` successfully. Using the randomness r for encrypting $\mathsf{pk}_\mathcal{U}$, this would have led to hid. Hence, any hidden user ID hid appearing within the view of a successful `SubmitReport` protocol run can be generated by an honest user.

The proofs for completeness with respect to `Collect` and `Redeem` work analogously, using the soundness of $P2$ and $P3$.

Extractability

The trapdoor td contains the secret decryption key sk_τ corresponding to pk_τ. For an honest user, hid is the encryption of the user's $\mathsf{pk}_\mathcal{U}$ under pk_τ. Hence, we can set $\texttt{ExtractUID}(\mathsf{td}, \mathsf{hid}) := E_\tau.\mathsf{Dec}(\mathsf{sk}_\tau, \mathsf{hid})$, which outputs $\mathsf{pk}_\mathcal{U}$. This works with respect to all three protocols in question: `SubmitReport`, `Collect` and `Redeem`.

7.6.4 Owner-binding

Theorem 7.15 (Owner-binding) *If the* Co-CDH *assumption holds and* $P1$*,* $P2$*,* $P3$*,* $P4$ *and* $P5$ *are perfectly* $F_{gp}^{(1)}$*-,* $F_{gp}^{(2)}$*-,* $F_{gp}^{(3)}$*-,* $F_{gp}^{(4)}$*- and* $F_{gp}^{(5)}$*-extractable, respectively,* $C_{\mathcal{I}}$*,* $C_{\mathcal{SP}}$ *and* $C_{\mathcal{V}}$ *are* $F_{gp}^{\mathcal{I}}$*,* $F_{gp}^{\mathcal{SP}}$ *and* $F_{gp}^{\mathcal{V}}$ *binding, respectively,* S *is* EUF-CMA *secure and* E_τ *is correct, then* GI *is* owner-binding*.*

The owner binding property is split into multiple experiments which are each required to only return one for any adversary with negligible probability. In the following, we give separate proofs for each of the experiments.

Owner-binding wrt. `Issue` and `RegisterUser`

Lemma 7.16 (Owner-binding wrt. `Issue` and `RegisterUser`) *If the Co-CDH assumption holds and* $P1$ *and* $P4$ *are perfectly* $F_{gp}^{(1)}$*- and* $F_{gp}^{(4)}$*-extractable, respectively, then* GI *is* owner-binding *with respect to* `Issue` *and* `RegisterUser`*.*

Proof Idea The `Issue` protocol is the same as in BBA+, wherefore the same proof idea holds. Our definition also covers the `RegisterUser` protocol. However, the changes made to the included issue protocol do not affect the proof (cf. [Har+19]).

Due to the extractability property of $P1$ and $P4$, $g_2^{\mathsf{sk}_{\mathcal{U}}}$ can be extracted from the proofs given during the `Issue` and `RegisterUser` protocols. This value is a solution to the Co-CDH instance $g_1, g_2, g_1^{\mathsf{sk}_{\mathcal{U}}} = \mathsf{pk}_{\mathcal{U}}$.

Owner-binding wrt. `SubmitReport`

Lemma 7.17 (Owner-binding wrt. `SubmitReport`) *If* $P4$ *and* $P5$ *are perfectly* $F_{gp}^{(4)}$*- and* $F_{gp}^{(5)}$*-extractable, respectively,* $C_{\mathcal{SP}}$ *is* $F_{gp}^{\mathcal{SP}}$*-binding,* S *is* EUF-CMA *secure and* E_τ *is correct, then* GI *is* owner-binding *with respect to* `Collect` *and* `Redeem`*.*

Proof idea The proof works analogously to the *owner-binding* with respect to `Accum` and `Verify` proof of BBA+ (cf. [Har+19]). Consider the first call to MalSubmitReport that fulfills the winning condition, ie. the adversary did call MalSubmitReport successfully for an extracted public key $\mathsf{pk}'_{\mathcal{U}}$ for which there has been no successful call to MalRegisterUser before. The soundness of the NIZK proof, the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$ and the correctness of E_τ ensure that the public key $\mathsf{pk}_{\mathcal{U}}$ extracted from the corresponding NIZK proof and the key $\mathsf{pk}'_{\mathcal{U}}$ extracted by `ExtractUID`(td, hid) are well-defined and identical.

Now, there are two cases to be distinguished. Let com' be the commitment from the NIZK proof. Either com' is a fresh commitment or it is a replayed commitment from a previous protocol invocation.

In the first case, the EUF-CMA security of the signature scheme S would be violated, as proofs only verify for signed commitments and the commitment has not been signed under $\mathsf{sk}_{\mathcal{SP}}$ before. In the second case, the adversary has to have equivocated an old commitment for some $\widehat{\mathsf{pk}}_{\mathcal{U}}$ to $\mathsf{pk}_{\mathcal{U}}$. This contradicts the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$.

Owner-binding wrt. `Collect` and `Redeem`

Lemma 7.18 (Owner-binding wrt. `Collect` and `Redeem`) *If $P1$, $P2$ and $P3$ are perfectly $F_{gp}^{(1)}$-, $F_{gp}^{(2)}$- and $F_{gp}^{(3)}$-extractable, respectively, $C_{\mathcal{I}}$ is $F_{gp}^{\mathcal{I}}$-binding, S is* EUF-CMA *secure and E_τ is correct, then* GI *is* owner-binding *with respect to* `Collect` *and* `Redeem`.

Proof idea The proof works analogously to the proof for Lemma 7.17, ie. the proof idea from BBA+ still holds (cf. [Har+19]).

Owner-binding bulletin mechanism

Lemma 7.19 (Owner-binding bulletin mechanism) *If $P3$ and $P5$ are perfectly $F_{gp}^{(3)}$- and $F_{gp}^{(5)}$-extractable, respectively, $C_{\mathcal{I}}$, $C_{\mathcal{SP}}$ and $C_{\mathcal{V}}$ are $F_{gp}^{\mathcal{I}}$, $F_{gp}^{\mathcal{SP}}$ and $F_{gp}^{\mathcal{V}}$ binding, respectively, and E_τ is correct, then* GI *has an* owner-binding *bulletin mechanism.*

Proof idea We adapt the proof idea for Lemma 7.17. Consider the first call to MalCollect that fulfills the winning condition, ie. the adversary did collect incentives for a $(\mathsf{pk}_{\mathcal{U}}, com_{\mathcal{V}})$ which has not been used within a successful call to `SubmitReport` before. However, there must have been a successful call to `SubmitReport` for $com_{\mathcal{V}}$ for a different user, ie. $(\widehat{pk}_{\mathcal{U}}, com_{\mathcal{V}}) \in \mathfrak{B}$, as otherwise $com_{\mathcal{V}}$ would not have been on the bulletin board and, therefore, could not have been collected using MalCollect. The soundness of $P5$, the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$, the $F_{gp}^{\mathcal{V}}$-binding property of $C_{\mathcal{V}}$ and the correctness of E_τ ensure that the public key $\widehat{pk}'_{\mathcal{U}}$ extracted from the corresponding NIZK proof, $\widehat{pk}_{\mathcal{U}}$ and the public key $\mathsf{com}_{\mathcal{V}}$ can be opened to are identical.

In the MalCollect call, the soundness of $P3$, the $F_{gp}^{\mathcal{I}}$-binding property of $C_{\mathcal{I}}$, the $F_{gp}^{\mathcal{V}}$-binding property of $C_{\mathcal{V}}$ and the correctness of E_τ ensure that the public key $\mathsf{pk}'_{\mathcal{U}}$ extracted from the corresponding NIZK proof, $\mathsf{pk}_{\mathcal{U}}$ and the public key $\mathsf{com}_{\mathcal{V}}$ can be opened to are identical.

Therefore, the adversary has to have equivocated the commitment $com_{\mathcal{V}}$ for $\widehat{pk}_{\mathcal{U}}$ to $\mathsf{pk}_{\mathcal{U}}$. This contradicts the $F_{gp}^{\mathcal{V}}$-binding property of $C_{\mathcal{V}}$.

7.6.5 Limit-binding

Theorem 7.20 (Limit-binding) *If $P4$ and $P5$ are perfectly sound and perfectly $F_{gp}^{(4)}$- and $F_{gp}^{(5)}$-extractable, respectively, $C_{\mathcal{SP}}$ is $F_{gp}^{\mathcal{SP}}$-binding, S is* EUF-CMA *secure and E_τ is correct, then* GI *is* limit-binding.

Proof We adapt the proof idea for the *balance-binding* property of BBA+. The proof consists of a series of Game hops, starting with the original *limit-binding* experiment and modifying it until the adversary has no chance to win. If the adversary could distinguish between two consecutive games, we could use the adversary to construct a new adversary breaking the security of one of the building blocks of the protocol.

Let us only consider report counter tokens for a fixed query identity $\widehat{qid}$ and, therefore, a fixed mobile node registration value $\widehat{\mathsf{regMN}}_{qid}$ contained in $\tau_{\mathcal{SP}}$. Note that the probability that two query identities

have the same mobile node registration value is negligible. Note further that the first MalRegisterUser call of the adversary $\mathcal{A}$ also fixes the report limit $\widehat{max}_{qid} > 0$, because of the definition of the oracle. We denote the advantage of $\mathcal{A}$ in game i with $\mathsf{Adv}^{\text{LB-game-1}}_{\text{GI},\mathcal{A}}(n)$. The following proof works for any (qid, max_{qid}) pair.

Game 1

We start with the original *limit-binding* experiment $\mathsf{Exp}^{\text{LB}}_{\text{GI},\mathcal{A}}$ Additionally, the game keeps track of the internal details of all successful transactions from the SP's and ISP's point of view.

For every successful ($b_{\mathcal{RA}} = 1$) call to MalRegisterUser a record $(\mathsf{pk}_{\mathcal{U}}, \pi, \mathsf{com}_{\text{out}}, s'', CTR = g_1^{\widehat{max}_{qid}})$ is stored. Hereby, $\mathsf{pk}_{\mathcal{U}}$ is the user's public key, given as an input to the RA, π is the proof sent by $\mathcal{U}$ to the $\mathcal{RA}$ and $com_{\text{out}} = com$ and s'' are the commitment and random value sent to $\mathcal{U}$ by the RA. CRT is the G_1 element representing $\widehat{max}_{qid}$, which is the value crt should be initialized with during `RegisterUser`. We use its G_1 element, as we want to compare it with values we can extract from the proofs received during calls of the MalSubmitReport oracle.

For every successful ($b_{\mathcal{I}} = 1$) call to MalSubmitReport a record $(\mathsf{pk}_{\mathcal{U}}, \pi, \mathsf{com}_{\text{in}}, \mathsf{com}_{\text{out}}, s, s'', CTR)$ is stored. Hereby, $\mathsf{pk}_{\mathcal{U}}$ is the public key extracted with `ExtractUID`, π and s are the proof and the token version number send by $\mathcal{U}$, $\mathsf{com}_{\text{in}} = com$ is the commitment extracted from π, $com_{\text{out}} = com^*$ and s'' are the commitment and random value send to $\mathcal{U}$ by the SP and CTR is the current report counter as a G_1-element which can be extracted from π as $P5$ is perfectly extractable.

By definition, we have that

$$\mathsf{Adv}^{\text{LB-game-1}}_{\text{GI},\mathcal{A}}(n) = \mathsf{Adv}^{\text{LB}}_{\text{GI},\mathcal{A}}(n) \tag{7.1}$$

Game 2

We first ensure that the submitted reports contain the correct mobile node registration value $\widehat{\mathsf{regMN}}_{qid}$. Therefore, we modify the experiment as follows: After every successful `SubmitReport` transaction, where $c = (\mathsf{regMN}^*_{qid}, c_1)$ has been submitted, the experiment verifies that $\mathsf{regMN}^*_{qid} \stackrel{?}{=} \widehat{\mathsf{regMN}}_{qid}$. If this is not the case, then the experiment aborts and returns 0. We call this event failure event F_1 (wrong query identity).

In case F_1 occurs, let $\widehat{rec} := (\widehat{pk}_{\mathcal{U}}, \widehat{\pi}, \widehat{com}_{\text{in}}, \widehat{com}_{\text{out}}, \widehat{s}, \widehat{s''}, \widehat{CTR})$ denote the record on a new successful MalSubmitReport call. A signature σ on the new message $(\mathsf{regMN}^*_{qid}, \widehat{com}_{\text{in}})$ can be extracted from $\widehat{\pi}$, which has to be valid due to the soundness of $P5$. Hence, in this case we can construct an adversary $\mathcal{B}_0$ against the EUF-CMA security of S with the advantage

$$\mathsf{Adv}^{\text{EUF-CMA}}_{S,\mathcal{B}_0}(n) = \Pr[F_1] \tag{7.2}$$

Game 3

Let us picture the set of successful transactions as a directed graph. For each transaction, there is

a node within the graph labeled with the corresponding record. There is an edge from node A to node B if A's com_{out} label equals B's com_{in} label.

For the third game, we modify the second game as follows: On every successful MalSubmitReport call, the experiments verifies that there exists a predecessor MalRegisterUser or MalSubmitReport record. More precisely, let $\widehat{rec} := (\widehat{pk}_{\mathcal{U}}, \widehat{\pi}, \widehat{com}_{\mathsf{in}}, \widehat{com}_{\mathsf{out}}, \widehat{s}, \widehat{s}'', \widehat{CTR})$ denote the record on a new successful MalSubmitReport call. If no previous record satisfying $com_{\mathsf{out}} = \widehat{com}_{\mathsf{in}}$ exists, then the experiment aborts and returns 0. We call this event failure event F_2 (no predecessor commitment).

Note that if F_2 occurs, a signature σ for the new message $(\widehat{\mathsf{regMN}}_{qid}, \widehat{com}_{\mathsf{in}})$ can be extracted from $\widehat{\pi}$, which has to be valid due to the soundness of $P5$. Hence, in this case we can construct an adversary $\mathcal{B}_1$ against the EUF-CMA security of S with the advantage

$$\mathsf{Adv}^{\text{EUF-CMA}}_{S,\mathcal{B}_1}(n) = \Pr[F_2] \tag{7.3}$$

In case F_2 does not occur, every commitment com_{in} used within a MalSubmitReport call has been generated in a previous transaction, which means that in the transaction graph, the indegree of every node representing a MalSubmitReport transaction is at least one.

Game 4

In the fourth game, the experiment additionally verifies that the indegree of every node in the transaction graph is at most one. More precisely, the experiment additionally checks whether $\widehat{com}_{\mathsf{in}}$ occurs in more than one previous transaction as com_{out}. If this is the case, the experiment aborts and returns 0. We call this event failure event F_3 (two predecessor commitments).

Let rec_0 and rec_1 be the record of two such predecessor transactions, containing the random numbers s_0'' and s_1'', which have been sent to $\mathcal{U}$ during these transactions. We split the event into two subevents $F_3^{(s_0''=s_1'')}$ and $F_3^{(s_0''\neq s_1'')}$ based on whether s_0'' and s_1'' are identical.

Note that s_0'' and s_1'' have been chosen uniformly at random from $\mathbb{Z}_p$ by the SP. Therefore, for $F_3^{(s_0''=s_1'')}$, we have that

$$\Pr\left[F_3^{(s_0''=s_1'')}\right] := \Pr\left[F_3 \wedge s_0'' = s_1''\right] \leq \frac{m^2}{p} \tag{7.4}$$

where m is the polynomial bound on the number of MalSubmitReport queries.

For $F_3^{(s_0''=s_1'')}$, we can extract two valid openings for $\widehat{com}_{\mathsf{in}}$ for the proofs included in rec_0 and rec_1 such that the corresponding implicit message vectors differ in the token version number component. If this case occurs, we can construct an adversary C_0 against the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$ with the advantage

$$\mathsf{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_0}(n) = \Pr\left[F_3^{(s_0''=s_1'')}\right] \tag{7.5}$$

Note that in case F_3 does not occur, the indegree of every node representing a MalSubmitReport

transaction is exactly one.

Game 5

We modify the previous game with an additional check if $\widehat{com}_{\text{in}}$ has already occurred as com_{in} in a previous transaction. If this is the case, the experiment aborts and returns 0. We call this event failure event F_4 (two successor commitments).

Let rec be such a record with the commitment $com_{\text{in}} = \widehat{com}_{\text{in}}$ extracted from the proof π and s being the token version number send by $\mathcal{U}$. We distinguish two different subevents, $F_4^{(s=\widehat{s})}$ and $F_4^{(s\neq\widehat{s})}$, based on whether $s = \widehat{s}$.

For $F_4^{(s=\widehat{s})}$, due to the winning condition of $\mathcal{A}$ excluding this case, we have that

$$\Pr\left[\text{Exp}^{\text{LB-game-1}}_{\text{GI},\mathcal{A}}(n) = 1 \wedge F_4^{(s=\widehat{s})}\right] = 0 \tag{7.6}$$

In case $F_4^{(s\neq\widehat{s})}$ occurs, we can extract two implicit messages $M \neq \widehat{M}$ and opening values $d, \widehat{d}$ from π and $\widehat{\pi}$ for $com_{\text{in}} = \widehat{com}_{\text{in}}$, respectively. Therefore, we can construct an adversary C_1 against the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$ with the advantage

$$\text{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_1}(n) = \Pr\left[F_4^{(s\neq\widehat{s})}\right] \tag{7.7}$$

If F_4 does not occur, every node in the transaction graph has an outdegree of at most 1.

Game 6

In the sixth game, the experiment verifies for every MalSubmitReport node that the same public key has been used as within its predecessor node. More precisely, the experiment additionally checks for each new MalSubmitReport call $\widehat{rec}$ if for the previous record where $\widehat{com}_{\text{in}}$ occurred as com_{out} (there is exactly one by now) it also holds that the included user public key $\text{pk}_{\mathcal{U}} = \widehat{pk}_{\mathcal{U}}$. If this is not the case, the experiment aborts and returns 0. We call this even failure event F_5 (token sharing).

Let rec be such a previous record containing $com_{\text{out}} = \widehat{com}_{\text{in}}$ but $\text{pk}_{\mathcal{U}} \neq \widehat{\text{pk}_{\mathcal{U}}}$ and the proof π. As $P4$ and $P5$ are sound and extractable and E_τ is correct and, therefore, perfectly binding, two different implicit messages $M \neq \widehat{M}$ and opening values $d, \widehat{d}$ for $com_{\text{out}} = \widehat{com}_{\text{in}}$ can be extracted from π and $\widehat{\pi}$. Therefore, an adversary C_2 on the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$ can be constructed with the advantage

$$\text{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_2}(n) = \Pr[F_5] \tag{7.8}$$

Note that so far, if no failure event occurs, then each MalSubmitReport node in the transaction graph has exactly one predecessor and at most one successor, which are both associated with the same user ID as the node itself. As there can be at most one successful register user call per $\text{pk}_{\mathcal{U}}$ by the definition of MalRegisterUser and the outdegree of the corresponding node is also limited

to at most one, there is only one such path per user ID. So to verify that the reporting limit has not been exceeded, we can check the length of these paths.

Game 7

In the seventh game, we verify that the counter values stored in the report limitation tokens are correct alongside the path, ie. the report counter value is decreased by one with each step along the path. On each new `SubmitReport` transaction, we check for the corresponding record $\widehat{rec}$ and it's predecessor record rec with $com_{\text{out}} = \widehat{com}_{\text{in}}$ and the extracted counter value CTR (as G_1 element) it holds that $\widehat{CTR} = CTR \cdot g_1^{-1}$ in case that rec is a record for a MalSubmitReport oracle call. If rec is a MalRegisterUser record, we check whether $\widehat{CTR} = CTR$. If the check fails, the experiment aborts and returns 0. We call this event failure event F_6 (wrong claim).

If F_6 occurs, we can extract two openings for $com_{\text{out}} = \widehat{com}_{\text{in}}$ from π and $\widehat{\pi}$ with different (implicit) report counter values. Therefore, we can construct an adversary C_3 on the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$ with the advantage

$$\mathsf{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_3}(n) = \Pr[F_6] \tag{7.9}$$

Game 8

At the end of the experiment, before returning 1 to $\mathcal{A}$ we now additionally check for each $\mathsf{pk}_{\mathcal{U}}$ used within the experiment that there are no more than $\widehat{max}_{qid} + 1$ (1 MalRegisterUser + $\widehat{max}_{qid}$) MalSubmitReport calls) successful transactions associated with $\mathsf{pk}_{\mathcal{U}}$. If the check fails, we return 0 instead. We call this event failure event F_7.

In case F_7 occurs, because each successful transaction is part of the unique path of $\mathsf{pk}_{\mathcal{U}}$ within the transaction graph, there has to be a path longer than $\widehat{max}_{qid} + 1$. As the for the first and second node in the path, we have that $CTR = g_1^{\widehat{max}_{qid}}$ and each subsequent step reduces the counter by one, in step $\widehat{max}_{qid} + 2$ it holds that $CTR = g_1^0 = 1$. As $P5$ is perfectly sound, we have that

$$\Pr[F_7] = 0 \tag{7.10}$$

Note that $\mathcal{A}$ cannot win Game 7.

Combining Equations (7.1) to (7.10), we obtain

$$
\begin{aligned}
\mathrm{Adv}^{\mathrm{LB}}_{\mathrm{GI},\mathcal{A}}(n) &= \mathrm{Adv}^{\text{LB-game-1}}_{\mathrm{GI},\mathcal{A}}(n) \\
&= \Pr\left[\mathrm{Exp}^{\text{LB-game-1}}(n) \stackrel{?}{=} 1 \wedge \neg\left(\bigvee_{i=1}^{7} F_i\right)\right] + \Pr\left[\mathrm{Exp}^{\text{LB-game-1}}(n) \stackrel{?}{=} 1 \wedge \left(\bigvee_{i=1}^{7} F_i\right)\right] \\
&= \mathrm{Adv}^{\text{LB-game-7}}_{\mathrm{GI},\mathcal{A}}(n) + \Pr\left[\mathrm{Exp}^{\text{LB-game-1}}(n) \stackrel{?}{=} 1 \wedge \left(\bigvee_{i=1}^{7} F_i\right)\right] \\
&\leq \Pr[F_1] + \Pr[F_2] + \Pr[F_3] + \Pr\left[\mathrm{Exp}^{\text{LB-game-1}}(n) \stackrel{?}{=} 1 \wedge F_4\right] + \Pr[F5] + \Pr[F_6] + \Pr[F_7] \\
&\leq \Pr[F_1] + \Pr[F_2] + \Pr\left[F_3^{s_0''=s_1''}\right] + \Pr\left[F_3^{(s_0''\neq s_1'')}\right] + \Pr\left[\mathrm{Exp}^{\text{LB-game-1}}(n) \stackrel{?}{=} 1 \wedge F_4^{(s=\widehat{s})}\right] \\
&\quad + \Pr\left[F_4^{s\neq\widehat{s}}\right] + \Pr[F5] + \Pr[F_6] + \Pr[F_7] \\
&\leq \mathrm{Adv}^{\text{EUF-CMA}}_{S,\mathcal{B}_0}(n) + \mathrm{Adv}^{\text{EUF-CMA}}_{S,\mathcal{B}_1}(n) + \frac{m^2}{p} + \mathrm{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_0}(n) + \mathrm{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_1}(n) \\
&\quad + \mathrm{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_2}(n) + \mathrm{Adv}^{F_{gp}^{\mathcal{SP}}\text{-binding}}_{C_{\mathcal{SP}},C_3}(n)
\end{aligned}
$$

as m is polynomial in n, S is EUF-CMA secure and $C_{\mathcal{SP}}$ is $F_{gp}^{\mathcal{SP}}$-binding, this advantage is negligible.

7.6.6 Balance-binding

Theorem 7.21 (Balance-binding) *If P1, P2 and P3 are perfectly $F_{gp}^{(1)}$-, $F_{gp}^{(2)}$- and $F_{gp}^{(3)}$-extractable, respectively, $C_{\mathcal{I}}$ is $F_{gp}^{\mathcal{I}}$-binding, S is* EUF-CMA *secure and E_τ is correct, then* GI *is* balance-binding.

Proof idea As our notion of *balance-binding* is essentially the same as within BBA+, the proof given in [Har+19] holds. The proof works analogously to the proof of Section 7.6.5 but storing the balance W and the accumulated/redeemed incentive values v instead of CTR in the records. Moreover, in Game 6, the experiment checks if the balance of the current record equals the balance contained within the previous record modified by v ($\widehat{W} = W \cdot g_1^v$ for `Collect`, $\widehat{W} = W \cdot g_1^{-v}$ for `Redeem`).Game 2 and Game 7 are not required, as the goal of the adversary is to claim a false balance and the balance tokens are independent of *qid*.

7.6.7 Double-spending detection

Theorem 7.22 (Double-spending detection) *If P1, P2, P3, P4 and P5 are perfectly $F_{gp}^{(1)}$-, $F_{gp}^{(2)}$-, $F_{gp}^{(3)}$-, $F_{gp}^{(4)}$- and $F_{gp}^{(5)}$-extractable, respectively, $C_{\mathcal{I}}$ and $C_{\mathcal{SP}}$ are additively homomorphic and $F_{gp}^{\mathcal{I}}$-binding and $F_{gp}^{\mathcal{SP}}$-binding, respectively, S is* EUF-CMA *secure and E_τ is correct, then* GI *ensures* double-spending detection.

Proof idea We adapt the proof idea from BBA+ to incorporate the report limitation mechanism (cf. [Har+19]). Let com_0, π_0 and com_1, π_1 denote the commitment and proof, and $s_0 = s_1$ the token version number contained in view_0 and view_1, respectively. Furthermore, let $z_0 = (t, u_2)$ and $z_1 = (t', u_2')$.

We consider each of the three winning conditions separately. For the first condition, we have that $\mathsf{pk}_{\mathcal{U}}^{(0)} \neq \mathsf{pk}_{\mathcal{U}}^{(1)}$. We distinguish three subcases:

$com_0 = com_1$
: Because of the correctness of E_τ and the soundness of $P2$, $P3$ and $P5$, this commitment can be opened using $\mathsf{pk}_{\mathcal{U}}^{(0)}$ and $\mathsf{pk}_{\mathcal{U}}^{(1)}$. Therefore, by the extractability property of the proofs, we can extract two valid, implicit messages $M^{(0)} \neq M^{(1)}$ and two openings $d^{(0)}$, $d^{(1)}$ for the commitment $com_0 = com_1$. This violates either the $F_{gp}^{\mathcal{I}}$-binding property of $C_{\mathcal{I}}$ or the $F_{gp}^{\mathcal{SP}}$-binding property of $C_{\mathcal{SP}}$.

$com_0 \neq com_1$ and com_0 or com_1 is new
: Either com_0 or com_1 did not occur in any previous transaction as a message sent from the $\mathcal{I}$ or $\mathcal{SP}$ and neither within a token issuing (via MalRegisterUser or MalIssue). In this case, $\mathcal{A}$ has managed to forge a signature in the name of the SP or ISP for this commitment, violating the EUF-CMA security of S.

$com_0 \neq com_1$ and com_0 and com_1 occurred before
: In this case, the commitments are already associated with some token version numbers s_0^* and s_1^*, which have been chosen uniformly at random. This is ensured as these token version numbers are generated by a Blum-like coin toss between the adversary and the ISP or SP. Even if the adversary cheats on his share s', adding the uniformly random s'' using the additive homomorphism of the corresponding commitment $C_{\mathcal{I}}$ or $C_{\mathcal{SP}}$ ensures that s is random. Therefore, the probability that the two version numbers generated by the protocols are equal is negligible. Hence, the adversary can equivocate at least one of the commitments for the winning condition $s_0 = s_1$ to occur with more than negligible probability, violating the binding property of the commitment.

For the second winning condition, we have that $\mathtt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1)) \neq (\mathsf{pk}_{\mathcal{U}}^{(0)}, \Pi)$. Let us consider $\mathsf{dstag}_0 = (s, (t, u_2))$ and $\mathsf{dstag}_1 = (s, (t', u_2'))$, where $t = \mathsf{sk}_{\mathcal{U}}^{(0)} u_2 + u_1$ and $t' = \mathsf{sk}_{\mathcal{U}}^{(1)} u_2' + u_1'$. Note that $\mathtt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1) = (\mathsf{pk}_{\mathcal{U}}^{(0)}, \mathsf{sk}_{\mathcal{U}}^{(0)})$ if the following conditions are satisfied: $\mathsf{sk}_{\mathcal{U}}^{(0)} = \mathsf{sk}_{\mathcal{U}}^{(1)}$, $\mathsf{pk}_{\mathcal{U}}^{(0)} = g_1^{\mathsf{sk}_{\mathcal{U}}^{(0)}}$, $u_2 \neq u_2'$ and $u_1 = u_1'$. We show that these conditions are satisfied with overwhelming probability, implying that this case can only occur with negligible probability.

We can assume that $\mathsf{pk}_{\mathcal{U}}^{(0)} = \mathsf{pk}_{\mathcal{U}}^{(1)}$ as otherwise we would be in the first case. Considering π_0 and π_1, from the soundness of $P2$, $P3$ and $P5$, the correctness of E_τ and the languages used within the proofs, it follows that $\mathsf{pk}_{\mathcal{U}}^{(0)} = g_1^{\mathsf{sk}_{\mathcal{U}}^{(0)}}$, $\mathsf{pk}_{\mathcal{U}}^{(1)} = g_1^{\mathsf{sk}_{\mathcal{U}}^{(1)}}$ and $\mathsf{sk}_{\mathcal{U}}^{(0)} = \mathsf{sk}_{\mathcal{U}}^{(1)}$. Moreover, as u_2 and u_2' are chosen uniformly at random from the ISP or SP, $u_2 \neq u_2'$ holds with overwhelming probability. The remaining possibility to violate the conditions of IdentDS would be if $u_1 \neq u_1'$. We distinguish between three subcases based on the commitments com_0 and com_1, similar to the first case.

$com_0 = com_1$
: The extractability of $P2$, $P3$ and $P5$ allows us to extract two implicit messages M_0 and M_1 and corresponding opening values for this commitment from π_0 and π_1. M_1 includes $g_1^{u_1}$ and M_2

includes $g_1^{u_1'}$. Therefore, $u_1 \neq u_1'$ violates either the $F_{gp}^{\mathcal{I}}$ property of $C_{\mathcal{I}}$ or the $F_{gp}^{\mathcal{SP}}$ property of $C_{\mathcal{SP}}$ based on whether $com_0 = com_1$ is a commitment for the incentive or the report limitation mechanism.

$com_0 \neq com_1$ and com_0 or com_1 is new
: In case either of the commitments has not occurred within a previous transaction, the EUF-CMA security of S would be violated as the extractability and soundness of the proof systems would allow the extraction of a valid signature for the new commitment from the corresponding proof.

$com_0 \neq com_1$ and com_0 and com_1 occurred before
: In this case, the token version numbers generated by the protocol runs are only equal with negligible probability. Hence, the adversary has to be able to equivocate at least one of the commitments com_0 and com_1 for $s_0 = s_1$ to hold with more than negligible probability, violating the binding property of the corresponding commitment.

For the third and last winning condition, we have that $\texttt{IdentDS}(\mathsf{pk}_{\mathcal{I}}, \mathsf{dstag}_0, \mathsf{dstag}_1)) = (\mathsf{pk}_{\mathcal{U}}^{(0)}, \Pi)$ but $\texttt{VerifyGuilt}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}^{(0)}, \Pi) = 0$. From our definition of `IdentDS` and `VerifyGuilt`, it follows that this case cannot occur.

7.6.8 Verifiable delivery

Theorem 7.23 (Verifiable delivery) *If* PI *is* node private *and* M *is* EUF-CMA secure*, then* GI *provides verifiable delivery.*

Proof idea To win the verifiable delivery experiment, $\mathcal{A}$ has to provide a valid MAC tag t_{mac} for $com_{\mathcal{V}} \| r_{\mathcal{V}}$ under the message authentication key k_{mac}. If $\mathcal{A}$ would be able to do this without the knowledge of k_{mac}, the EUF-CMA security of M would be violated.

Let us assume there exists an adversary $\mathcal{A}$ that can successfully extract the message authentication key k_{mac} from the experiment. We use this adversary to construct an adversary $\mathcal{A}'$ on the *node privacy* of PI. $\mathcal{A}'$ simulates the *verifiable delivery* experiment for $\mathcal{A}$ replacing the RA's public key $\mathsf{pk}_{\mathcal{RA}}$ with the key provided by the challenger. It then generates two message authentication keys $\mathsf{k}_{\text{mac}}^{(0)}$ and $\mathsf{k}_{\text{mac}}^{(1)}$ and chooses a random query identity and message qid, m. It calls its CorruptSP oracle and sends $((qid, m \| \mathsf{k}_{\text{mac}}^{(0)}), (qid, m \| \mathsf{k}_{\text{mac}}^{(1)}))$ to the challenger. The mobile node registration number contained in the challenge report $c = (\mathsf{regMN}_{qid}, c_1)$ is used within the simulation of `RegisterUser` and c is then injected into the `SubmitReport` protocol, replacing the one that would otherwise have been internally computed. This removes the need to know $RAsk$ and regMN_{qid}.

The MalRegisterUser and MalRegisterQ oracles available to $\mathcal{A}$ can be simulated by $\mathcal{A}'$ using its CorruptMN and CorruptQ oracles. As $\mathcal{A}$ does not know qid there is only a negligible possibility that he registers a user identity for this query identity, which would result in $\mathcal{A}'$ loosing in its experiment. In fact, if $\mathcal{A}$ would be able to extract the query identity from c, we could similarly use this to break the *node privacy* of PI. Note that $\mathcal{A}'$ could also use its knowledge of regMN_{qid} to simulate the oracle.

$\mathcal{A}'$ exploits the fact the subscription token st_{qid_b} is equivalent to the mobile node registration value regMN_{qid} contained in the report $c = (\mathsf{regMN}_{qid}, c_1)$ to simulate the execution of the `CollectReports` protocol. To detect modification, $\mathcal{A}'$ compares the report c^* returned together with $com_{\mathcal{V}}$ with c. Note that we use an ANO-IND-ID-CCA *secure* IBE scheme within PI which is therefore not homomporphic.

If $\mathcal{A}$ is successful in extracting $\mathsf{k}_{\mathrm{mac}}$, $\mathcal{A}'$ can compare it with $\mathsf{k}_{\mathrm{mac}}^{(0)}$ and $\mathsf{k}_{\mathrm{mac}}^{(1)}$ and return the according bit b' to its challenger to win the *node privacy* experiment.

7.6.9 Transaction unlinkability

Theorem 7.24 (Transaction unlinkability) *If* $P1$, $P2$, $P3$, $P4$ *and* $P5$ *are* composable zero-knowledge, $C_{\mathcal{I}}$, $C_{\mathcal{SP}}$ *and* $C_{\mathcal{V}}$ *are equivocable,* E_τ *is* IND-CPA *secure,* PI *provides* report unlinkability *and the VRF* F *used within* PI *is* correct *and* unique*, then* GI *provides* transaction unlinkability.

Proof The proof follows a series of game hops, similar to the *privacy-preserving* proof of BBA+ [Har+19]. We start from the real world experiment $\mathrm{Exp}^{\text{TU-real}}_{\mathrm{GI},\mathcal{A}}$ and modify it in several steps until we reach $\mathrm{Exp}^{\text{TU-ideal}}_{\mathrm{GI},\mathcal{A}}$. Hereby, we denote the experiments of the ith intermediate game by $\mathrm{Exp}^{\text{TU-game-i}}_{\mathrm{GI},\mathcal{A}}$. We further denote the oracles available to $\mathcal{A}$ in $\mathrm{Exp}^{\text{TU-game-i}}_{\mathrm{GI},\mathcal{A}}$ with $\mathrm{HonIssue}_i$, $\mathrm{HonRegisterUser}_i$, $\mathrm{HonSubmitReport}_i$, $\mathrm{HonCollect}_i$, $\mathrm{HonRedeem}_i$ and $\mathrm{Corrupt}_i$. $\mathtt{Setup}_i$ denotes the implementation of the setup algorithm used at the beginning of the experiment to generate the CRS. Note that the adversary additionally has access to HonUser, which remains unchanged between the games.

Game 1

We start with $\mathrm{Exp}^{\text{TU-real}}_{\mathrm{GI},\mathcal{A}}(n)$, that is, in Game 1 we set

$\mathtt{Setup}_1 := \mathtt{Setup}$
$\mathrm{HonIssue}_1 := \mathrm{RealHonIssue}$
$\mathrm{HonRegisterUser}_1 := \mathrm{RealHonRegisterUser}$
$\mathrm{HonSubmitReport}_1 := \mathrm{RealHonSubmitReport}$
$\mathrm{HonCollect}_1 := \mathrm{RealHonCollect}$
$\mathrm{HonRedeem}_1 := \mathrm{RealHonRedeem}$
$\mathrm{Corrupt}_1 := \mathrm{RealCorrupt}$

Game 2

In the second game, we modify $\mathtt{Setup}_2$ such that we generate all the CRSs for the zero-knowledge proofs and commitments in a way that we can simulate the corresponding proofs and equivocate the corresponding commitments, respectively. The full algorithm is given in Figure 7.48

As $P1$ to $P5$ are *composable zero-knowledge* and $C_{\mathcal{I}}$, $C_{\mathcal{SP}}$ and $C_{\mathcal{V}}$ are *equivocable*, $\mathcal{A}$ can distinguish between Game 1 and Game 2 with at most negligible advantage.

Game 3

In the third game, we exchange all the zero-knowledge proofs within $\mathrm{HonIssue}_3$, $\mathrm{HonRegisterUser}_3$, $\mathrm{HonSubmitReport}_3$, $\mathrm{HonCollect}_3$ and $\mathrm{HonRedeem}_3$ with simulated proofs using the corresponding proof simulators $\mathtt{SimProof}'$ (cf. Section 3.11). Note that in opposite to $\mathtt{SimProof}$, $\mathtt{SimProof}'$ takes the witness and returns $\perp$ if the statement and witness pair is not in the language.

$\mathtt{Setup}_2(1^n)$

$gp := (G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \mathtt{SetupGrp}(1^n)$
$\mathsf{hk} \leftarrow \mathtt{HGen}(1^n)$
$(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{I}}}) \leftarrow C_{\mathcal{I}}.\mathtt{SimGen}(gp)$
$(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{SP}}}) \leftarrow C_{\mathcal{SP}}.\mathtt{SimGen}(gp)$
$(\mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{V}}}) \leftarrow C_{\mathcal{V}}.\mathtt{SimGen}(gp)$
$(\mathsf{sk}_{\tau}, \mathsf{pk}_{\tau}) \leftarrow E_{\tau}.\mathtt{Gen}(gp)$
$(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{I}}}) \leftarrow \mathtt{SetupSPoK}(gp, (\mathsf{td}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{V}}})$
$(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{SP}}}) \leftarrow \mathtt{SetupSPoK}(gp, \mathsf{td}_{\mathsf{com}_{\mathcal{SP}}})$
$\mathsf{CRS}_{\mathsf{pok}_{\mathrm{PI}}} \leftarrow \mathrm{PI}.\mathtt{SetupVRF}(gp)$
$\mathsf{CRS} := (gp, \mathsf{hk}, \mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}}, \mathsf{pk}_{\tau}, \mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, \mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, \mathsf{CRS}_{\mathsf{pok}_{\mathrm{PI}}})$
$\mathsf{td}_{\mathsf{Sim}} := (\mathsf{sk}_{\tau}, \mathsf{td}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{V}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{SP}}})$
return $(\mathsf{CRS}, \mathsf{td}_{\mathsf{Sim}})$

Figure 7.48: $Setup_2$ algorithm

To show that Game 2 and Game 3 are indistinguishable, we use a nested game proof argument. First, note that the statements that are proven are correct and, therefore, $\mathtt{SimProof}'$ never returns $\perp$ but always a simulated proof.

In Game 3.1, we only replace the proofs for $P1$ with simulated proofs. We now assume there exists an adversary $\mathcal{A}'$ that can distinguish between Game 3.1 and Game 2 with non-negligible advantage. We construct an adversary $\mathcal{A}''$ against the *composable zero-knowledge* property of $P1$ as follows: $\mathcal{A}''$ runs $\mathcal{A}'$ internally and simulates the experiment and all the user oracles. However, $\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}$ is replaced with the CRS from the challenger and all internal calls to $P1.\mathtt{Prove}$ are forwarded to A'''s oracle that either returns a real or simulated proof. Finally, $\mathcal{A}''$ outputs the same decision as $\mathcal{A}'$. Note that $\mathcal{A}''$ has the same advantage as $\mathcal{A}'$.

Game 3.2 to 3.5 work analogously by replacing all proofs within $P2$ to $P5$, one proof system at a time. In the proof, when replacing $P4$ and $P5$, $\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}$ has to be replaced with the CRS from the challenger. Note that in Game 3.4 (where we replace $P4$), $\mathcal{A}'$ is further restricted as he only has access to the transcript of the interaction. Moreover, in any case, $\mathcal{A}''$ has to simulate all proofs for the previously replaced proof systems. This is possible as $\mathcal{A}''$ gets the simulation trapdoor $\mathsf{td}_{\mathsf{spok}}$ in the zero-knowledge game.

Game 4

We now replace all commitments with simulated commitments. If a user gets corrupted, the commitments are equivocated to contain the correct user secret key, balance or counter value and query identity, whichever is applicable for the corresponding commitment scheme. For the token version number s and the user randomness u_1 in $C_{\mathcal{I}}$ and $C_{\mathcal{SP}}$, new random $\mathbb{Z}_p$ elements are chosen. Additionally, before calling $\mathtt{UVer}_{\mathcal{SP}}$ or $\mathtt{UVer}_{\mathcal{I}}$, the corresponding commitment is equivocated to

contain the user secret key $\mathsf{sk}_{\mathcal{U}} := 0$, a query identity $qid := 0$, if applicable, and a known balance or counter value such that it is still possible to verify if $\mathcal{A}$ modified the commitments correctly. As the statements proven with $P1$ to $P5$ are no longer valid, the corresponding proof simulator algorithms `SimProof` have to be used directly instead of the wrapper `SimProof'`.

Again, we show that Game 3 and Game 4 are indistinguishable by a nested game proof argument. More precisely, we use two levels of nested arguments. We first define the games Game 4.1, 4.2, and 4.3 in which we replace all the usage of $C_{\mathcal{I}}$, $C_{\mathcal{SP}}$, and $C_{\mathcal{V}}$, respectively. We only give a proof for the indistinguishability of Game 3 and Game 4.1 as the two remaining steps work analogously.

To show the indistinguishability of Game 4.1 and Game 3, we assume there exists an adversary $\mathcal{A}'$ distinguishing those games with non-negligible advantage. We define the games Game 4.1.i for $i \in [0,m]$ as follows: In Game 4.1.i, the first i $C_{\mathcal{I}}$ commitments that are sent to $\mathcal{A}$ are replaced with simulated commitments generated with $C_{\mathcal{I}}$.`SimGen`. Hereby, Game 4.1.0 equals Game 3 and Game 4.1.m is equal to Game 4.1. As $\mathcal{A}'$ can distinguish Game 4.1.m and Game 4.1.0 there exists an index i such that $\mathcal{A}'$ can distinguish between Game 4.1.i and Game 4.1.$(i-1)$. We use this to construct an adversary $\mathcal{A}''$ on the *equivocability* of $C_{\mathcal{I}}$. $\mathcal{A}''$ runs $\mathcal{A}'$ internally and simulates the experiment and the oracles for $\mathcal{A}'$. As $\mathcal{A}''$ gets the commitment trapdoor $\mathsf{td}_{\mathsf{com}_{\mathcal{I}}}$ from the *equivocability* challenger, it can generate the first $i-1$ $C_{\mathcal{I}}$-commitments with $C_{\mathcal{I}}$.`SimGen` and equivocate for $\mathtt{UVer}_{\mathcal{I}}$ and Corrupt. Moreover, $\mathcal{A}''$ has the ith $C_{\mathcal{I}}$-commitment be generated by the challenger. All remaining $C_{\mathcal{I}}$-commitments are generated by $C_{\mathcal{I}}$.`Gen` as in the real world experiment. $\mathcal{A}''$ has the same advantage as $\mathcal{A}'$ has in distinguishing Game 4.1.i and 4.1.$(i-1)$.

Game 5

In the fifth game, we modify HonSubmitReport_5 as follows: Let $\mathsf{pk}_{\mathcal{U}}$ be the user identity and qid be the query identity used within the call. If there was at least one successful call to HonSubmitReport_5 for $\mathsf{pk}_{\mathcal{U}}$ and qid between this call and the last call to Corrupt_5 for $\mathsf{pk}_{\mathcal{U}}$ or the successful call to HonRegisterUser_5 for $\mathsf{pk}_{\mathcal{U}}$ and qid took place after Corrupt_5 has been called, then the value t, which is used during the corresponding interactive protocols as part of the double-spending detection mechanism, is chosen randomly and not calculated as $t := \mathsf{sk}_{\mathcal{U}} u_2 + u_1$.

Analogously, we modify HonCollect_5 and HonRedeem_5, by choosing t randomly if there was at least one successful HonCollect_5 or HonRedeem_5 call between this call and the last call to Corrupt_5 for the same user identity.

Excluding a transaction directly following a corruption of the user is necessary as in this case, $\mathcal{A}$ knows u_1, u_2 and $\mathsf{sk}_{\mathcal{U}}$ when getting t and can, therefore, check if t was created honestly. Note that we do not have to distinguish between HonCollect_5 and HonRedeem_5 calls as they use the same token.

For transactions not directly following a corruption where the corresponding token $\tau_{\mathcal{I}}^{\mathcal{U}}$ or $\tau_{\mathcal{SP}}^{\mathcal{U},qid}$ is exposed, $\mathcal{A}$ does not now u_1, which is chosen uniformly at random in every transaction in Game 4. Therefore, t looks uniformly random to $\mathcal{A}$ as it is the case in Game 5, which is why $\mathcal{A}$

cannot distinguish those two games.

Game 6

We replace the encryption hid of $\mathsf{pk}_{\mathcal{U}}$ within HonSubmitReport_6, HonCollect_6 and HonRedeem_6 with an encryption of $g_1^0 = 1$. However, we do not change the cleartext $\mathsf{pk}_{\mathcal{U}}$ used within the identifying protocols `Issue` and `RegisterUser`, as they are part of the proof statement and, therefore, the change could be easily detected by $\mathcal{A}$.

To show that this modification is indistinguishable from Game 5, we assume there exists an adversary $\mathcal{A}'$ distinguishing between Game 5 and Game 6 with non-negligible advantage. We define Game 6.0 to Game 6.m by replacing the encryptions one by one in the same order as they are sent to $\mathcal{A}$. Hereby, Game 6.0 equals Game 5 and Game 6.m equals Game 6. In Game 6.i, the first i hids are replaced with an encryption of 1 while the following are encryptions of the real $\mathsf{pk}_{\mathcal{U}}$. As $\mathcal{A}'$ can distinguish between Game 6.0 and Game 6.m, there exists an index i such that $\mathcal{A}'$ can distinguish between Game 6.i and Game 6.$(i-1)$. We use this to construct an adversary $\mathcal{A}''$ on the IND-CPA security of E_τ. Note that $\mathcal{A}'$ never gets the decryption key sk_τ. $\mathcal{A}''$ runs $\mathcal{A}'$ internally and simulates the experiment and the user oracles for $\mathcal{A}'$. Up to the $(i-1)$th hid, $\mathcal{A}''$ generates them as encryptions of 1. For the ith hid, $\mathcal{A}''$ sends $(\mathsf{pk}_{\mathcal{U}}, 1)$ to its challenger C and uses the encryption obtained for C. All following hids are generated as encryptions of $\mathsf{pk}_{\mathcal{U}}$. The decision of $\mathcal{A}''$ is the same as the decision of $\mathcal{A}'$. In this construction $\mathcal{A}''$ has the same advantage as $\mathcal{A}'$ has in distinguishing Game 6.i and Game 6.$(i-1)$.

Game 7

In Game 7, for all $(\mathsf{pk}_{\mathcal{U}}, qid)$ where the user identified by $\mathsf{pk}_{\mathcal{U}}$ has not previously been corrupted while already registered for qid, we replace the report counter token $\tau_{\mathcal{SP}}$ used within HonSubmitReport_7 with a $\tau^*_{\mathcal{SP}}$ randomly chosen from all the report counter tokens observed in previous successful calls to HonRegisterUser_7 for the same qid.

We assume there exists an adversary $\mathcal{A}'$ distinguishing between Game 6 and Game 7 with non-negligible advantage. Note that as we already simulate the proofs within `RegisterUser`, only the mobile node registration value from the report counter token is used to execute `SubmitReport`. Therefore, as long as all the mobile node registration values for the same query identity qid issued to users played by the experiment are equal, the modifications made in Game 7 are not observable by $\mathcal{A}'$. Let, therefore, $\mathsf{regMN}_{qid} \neq \mathsf{regMN}^*_{qid}$ be two different mobile node registration values send by $\mathcal{A}'$ during two successful `RegisterUser` protocol runs for the same query identity qid. Let further π_{qid} and π^*_{qid} be the corresponding VRF proofs send by $\mathcal{A}'$. As the user accepted the protocol run, it holds that $F.\texttt{Verify}(\mathsf{pk}_F, qid, \mathsf{regMN}_{qid}, \pi_{qid}) \stackrel{?}{=} F.\texttt{Verify}(\mathsf{pk}_F, qid, \mathsf{regMN}^*_{qid}, \pi^*_{qid}) \stackrel{?}{=} 1$, which is a direct contradiction to the *uniqueness* of F.

Figures 7.49 to 7.53 show the behavior of $\mathcal{U}_{\mathsf{Sim}}$ in Game 7 and Figure 7.54 the Sim algorithm used within Corrupt_7, accumulating the changes made in the previous games.

$\texttt{Issue}\left\langle \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})\right\rangle$

$\boldsymbol{\mathcal{U}}_{\mathbf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$	$\boldsymbol{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}})$
$s', u_1 \leftarrow \mathbb{Z}_p$	
$(com', r') \leftarrow C_{\mathcal{I}}.\mathtt{SimCom}(gp)$	
$stm := (com', \mathsf{pk}_{\mathcal{U}})$	
$\pi \leftarrow P1.\mathtt{SimProof}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{I}}}, stm)$	
$\xrightarrow{com', \pi}$	
	$stm := (com', \mathsf{pk}_{\mathcal{U}})$
	if $P1.\mathtt{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
	return 0
	endif
	$s'' \leftarrow \mathbb{Z}_p$
	$(com'', d'') \leftarrow C_{\mathcal{I}}.\mathtt{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s'', 0, 0, 0))$
	$com := com' \cdot com''$
	$\sigma \leftarrow S.\mathtt{Sign}(\mathsf{sk}_{\mathcal{I}}, com)$
$\xleftarrow{com, d'', \sigma, s''}$	
$s := s' + s'' \mod p$	
$d' \leftarrow C_{\mathcal{I}}.\mathtt{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{I}}}, (s, 0, 0, u_1), r')$	
$d := d' \cdot d''$	
$\tau := (com, d, \sigma, s, u_1)$	
if $\mathtt{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, 1.0.\tau, 0) \stackrel{?}{=} 0$ **then return** $(\perp, 0)$	
endif	
return $(\perp, 1)$	**return** 1

Figure 7.49: Behavior of $\mathcal{U}_{\mathsf{Sim}}$ during the `Issue` protocol in $\mathrm{HonIssue}_7$

$\texttt{RegisterUser}\ \langle \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \rangle$

$\boldsymbol{\mathcal{U}}_{\mathbf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid)$		$\boldsymbol{\mathcal{RA}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid})$
	$\xleftarrow{max_{qid}}$	
$s', u_1 \leftarrow \mathbb{Z}_p$		
$(com', d') \leftarrow C_{\mathcal{SP}}.\texttt{SimCom}(gp)$		
$stm \leftarrow (com', \mathsf{pk}_{\mathcal{U}}, g_1^{max_{qid}})$		
$\pi \leftarrow P4.\texttt{SimProof}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{SP}}}, stm)$		
	$\xrightarrow{com', \pi}$	
		$stm \leftarrow (com', \mathsf{pk}_{\mathcal{U}}, g_1^{max_{qid}})$
		if $P4.\texttt{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return 0
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{SP}}.Com(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, (s'', 0, 0, 0, 0))$
		$com := com' \cdot com''$
		$\mathsf{regMN}_{qid} \leftarrow \mathrm{PI}.\texttt{RegisterMN}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
	$\xleftarrow{\mathsf{regMN}_{qid}, com, d'', \sigma, s''}$	$\sigma \leftarrow S.\texttt{Sign}(\mathsf{sk}_{\mathcal{SP}}, com)$
$s := s' + s'' \mod p$		
$d' \leftarrow C_{\mathcal{SP}}.\texttt{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{SP}}}, (s, max_{qid}, 0, u_1), r')$		
$d := d' \cdot d''$		
$\tau_{\mathcal{SP}} := (com, d, \sigma, s, u_1, \mathsf{regMN}_{qid}, max_{qid})$		
if $\texttt{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, 1, 0, \tau_{\mathcal{SP}}) \stackrel{?}{=} 0$ **then**		
return $(\perp, \perp, 0)$		
endif		
return $(\tau_{\mathcal{SP}}, 1)$		**return** 1

Figure 7.50: Behavior of $\mathcal{U}_{\mathsf{Sim}}$ during the `RegisterUser` protocol in $\mathrm{HonRegisterUser}_7$

$\texttt{SubmitReport}\ \langle \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, qid, \tau^*_{\mathcal{SP}}, m), \mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}}) \rangle$

$\mathcal{U}_{\mathbf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{SP}}, qid, \tau^*_{\mathcal{SP}}, m)$		$\mathcal{SP}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{sk}_{\mathcal{SP}})$
	$\xleftarrow{u_2}$	$u_2 \leftarrow \mathbb{Z}_p$
$\mathsf{k}_{\mathrm{mac}} \leftarrow M.\mathsf{Gen}(1^n)$		
$(\widehat{com}, \widehat{d}, \widehat{\sigma}, \widehat{s}, \widehat{u_1}, \mathsf{regMN}_{qid}, \widehat{ctr}) := \tau^*_{\mathcal{SP}}$		
$c \leftarrow \mathrm{PI}.\mathsf{ReportData}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m \Vert \mathsf{k}_{\mathrm{mac}})$		
$(com_{\mathcal{V}}, d_{\mathcal{V}}) \leftarrow C_{\mathcal{V}}.\mathsf{SimCom}(gp)$		
$t \leftarrow \mathbb{Z}_p$		
$r, s', u'_1 \leftarrow \mathbb{Z}_p$		
$\mathsf{hid} := E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, 1; r)$		
$(com', r') \leftarrow C_{\mathcal{SP}}.\mathsf{SimCom}(gp)$		
$stm := (com', g_1^s, t, u_2, \mathsf{regMN}_{qid}, \mathsf{hid})$		
$\pi \leftarrow P5.\mathsf{SimProof}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{SP}}}, stm)$		
	$\xrightarrow{c, com_{\mathcal{V}}, com', s, t, \pi, \mathsf{hid}}$	
		$z := (t, u_2)$
		$\mathsf{dstag} := (s, z)$
		$(\mathsf{regMN}_{qid}, c_1) := c$
		$stm := (com', g_1^s, t, u_2, \mathsf{regMN}_{qid}, \mathsf{hid})$
		if $P5.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{SP}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return $(\bot, \bot, \bot, 0)$
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{SP}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, (s'', -1, 0, 0, 0))$
		$com^* := com' \cdot com''$
		$\sigma^* \leftarrow S.\mathsf{Sign}(\mathsf{sk}_{\mathcal{SP}}, \mathsf{regMN}_{qid}, com^*)$
	$\xleftarrow{com^*, d'', \sigma^*, s''}$	
$s^* := s' + s'' \mod p$		
$d' := C_{\mathcal{SP}}.\mathsf{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{SP}}}, (s', 1, 0, u'_1), r')$		
$d^* := d' \cdot d''$		
$u_1^* := u'_1$		
$\tau^*_{\mathcal{SP}} := (com^*, d^*, \sigma^*, s^*, u_1^*, \mathsf{regMN}_{qid}, 0)$		
if $\mathsf{UVer}_{\mathcal{SP}}(\mathsf{pk}_{\mathcal{SP}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau^*_{\mathcal{SP}}) \stackrel{?}{=} 0$ **then**		
return $(\bot, 0)$		
endif		
$\tau_{\mathcal{V}} := (com_{\mathcal{V}}, d_{\mathcal{V}})$		
return $(\tau_{\mathcal{V}}, \bot, 1)$		**return** $(c, \mathsf{com}_{\mathcal{V}}, \mathsf{dstag}, \mathsf{hid}, 1)$

Figure 7.51: Behavior of $\mathcal{U}_{\mathsf{Sim}}$ during the SubmitReport protocol in $\mathrm{HonSubmitReport}_7$

$\texttt{Collect}\left\langle \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, com_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}}), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{com}_{\mathcal{V}}, r_{\mathcal{V}}, v)\right\rangle$

$\mathcal{U}_{\mathbf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, com_{\mathcal{V}}, m)$		$\mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, \mathsf{com}_{\mathcal{V}}, r_{\mathcal{V}}, v)$
	$\xleftarrow{u_2, r_{\mathcal{V}}, v}$	$u_2 \leftarrow \mathbb{Z}_p$
$t \leftarrow \mathbb{Z}_p$		
$r, s', u_1' \leftarrow \mathbb{Z}_p$		
$\mathsf{hid} := E_{\tau}.\mathsf{Enc}(\mathsf{pk}_{\tau}, 1; r)$		
$(com', r') \leftarrow C_{\mathcal{I}}.\mathsf{SimCom}(gp)$		
$stm := (com', com_{\mathcal{V}}, g_1^s, t, u_2, hid)$		
$\pi \leftarrow P2.\mathsf{SimProof}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{I}}}, stm)$		
	$\xrightarrow{com', s, t, \pi, \mathsf{hid}}$	
		$z := (t, u_2)$
		$\mathsf{dstag} := (s, z)$
		$stm := (com', com_{\mathcal{V}}, g_1^s, t, u_2, \mathsf{hid})$
		if $P2.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return 0
		endif
		$s'' \leftarrow \mathbb{Z}_p$
		$(com'', d'') \leftarrow C_{\mathcal{I}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s'', v, 0, 0))$
		$com^* := com' \cdot com''$
		$\sigma^* \leftarrow S.\mathsf{Sign}(\mathsf{sk}_{\mathcal{I}}, com^*)$
	$\xleftarrow{com^*, d'', \sigma^*, s''}$	
$s^* := s' + s'' \mod p$		
$d' \leftarrow C_{\mathcal{I}}.\mathsf{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{I}}},$ $(s', 0, 0, u_1'), r')$		
$d^* := d' \cdot d''$		
$u_1^* := u_1'$		
$\tau_{\mathcal{I}}^* := (com^*, d^*, \sigma^*, s^*, u_1^*)$		
if $\mathsf{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, 1.0.\tau_{\mathcal{I}}^*, v) \stackrel{?}{=} 0$ **then**		
return $(\perp, 0)$		
endif		
$t_{\mathrm{mac}} \leftarrow M.\mathsf{Mac}(\mathsf{k}_{\mathrm{mac}}, com_{\mathcal{V}}, r_{\mathcal{V}})$	$\xrightarrow{t_{\mathrm{mac}}}$	
return $(\perp, 1, v)$		**return** $(t_{\mathrm{mac}}, \mathsf{dstag}, \mathsf{hid}, 1)$

Figure 7.52: Behavior of $\mathcal{U}_{\mathsf{Sim}}$ during the `Collect` protocol in $\mathrm{HonCollect}_7$

$\mathsf{Redeem}\left\langle \mathcal{U}_{\mathsf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, w_{\mathcal{U}}, v), \mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w_{\mathcal{U}}, v)\right\rangle$

$\mathcal{U}_{\mathbf{Sim}}(\mathsf{td}_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{I}}, w_{\mathcal{U}}, v)$		$\mathcal{I}(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}}, w, v)$
	$\xleftarrow{u_2}$	$u_2 \leftarrow \mathbb{Z}_p$
$t \leftarrow \mathbb{Z}_p$ $r, s', u_1' \leftarrow \mathbb{Z}_p$ $\mathsf{hid} := E_\tau.\mathsf{Enc}(\mathsf{pk}_\tau, 1; r)$ $(com', r') \leftarrow C_{\mathcal{I}}.\mathsf{SimCom}(gp)$ $stm := (com', g_1^s, t, u_2, \mathsf{hid}, g_1^{w_{\mathcal{U}}})$ $\pi \leftarrow P3.\mathsf{SimProof}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{spok}_{\mathcal{I}}}, stm)$		
	$\xrightarrow{com', s, t, \pi, \mathsf{hid}}$	
		$z := (t, u_2)$ $\mathsf{dstag} := (s, z)$ $stm := (com', g_1^s, t, u_2, \mathsf{hid}, g_1^{w_{\mathcal{U}}})$ **if** $P2.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{pok}_{\mathcal{I}}}, stm, \pi) \stackrel{?}{=} 0$ or $w < v$ **then** **return** $(\perp, \perp, 0)$ **endif** $s'' \leftarrow \mathbb{Z}_p$ $(com'', d'') \leftarrow C_{\mathcal{I}}.\mathsf{Com}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, (s'', -v, 0, 0))$ $com^* := com' \cdot com''$ $\sigma^* \leftarrow S.\mathsf{Sign}(\mathsf{sk}_{\mathcal{I}}, com^*)$
	$\xleftarrow{com^*, d'', \sigma^*, s''}$	

$s^* := s' + s'' \mod p$
$d' \leftarrow C_{\mathcal{I}}.\mathsf{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{I}}}, (s', v, 0.u_1'), r')$
$d^* := d' \cdot d''$
$u_1^* := u_1'$
$\tau_{\mathcal{I}}^* := (com^*, d^*, \sigma^*, s^*, u_1^*)$
if $\mathsf{UVer}_{\mathcal{I}}(\mathsf{pk}_{\mathcal{I}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \tau_{\mathcal{I}}^*, 0) \stackrel{?}{=} 0$ **then**
return $(\perp, 0)$
endif
return $(\perp, 1, -v)$
		return $(\mathsf{dstag}, \mathsf{hid}, 1)$

Figure 7.53: Behavior of $\mathcal{U}_{\mathsf{Sim}}$ during the Redeem protocol in $\mathrm{HonRedeem}_7$

$\mathsf{Sim}(td_{\mathsf{Sim}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, w_{\mathcal{U}}, \mathfrak{R}, \mathfrak{S}'_{\mathcal{U}}, \mathfrak{S}^*_{\mathcal{U}})$

Let $com^*_{\mathcal{I}}$ and $\sigma^*_{\mathcal{I}}$ be the latest commitment and signature that were send by $\mathcal{A}$ during HonCollect or HonRedeem calls for $\mathsf{pk}_{\mathcal{U}}$ and let $r'_{\mathcal{I}}$. Let further $com^*_{\mathcal{SP},qid}$ and $\sigma^*_{\mathcal{SP},qid}$ be the latest commitment and signature send by $\mathcal{A}$ during SubmitReport for $\mathsf{pk}_{\mathcal{U}}$ and qid.

$s^*, u_1^* \leftarrow \mathbb{Z}_p$
$d^* := C_{\mathcal{I}}.\mathtt{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{I}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{I}}}, (s^*, w, \mathsf{sk}_{\mathcal{U}}, u_1^*), r')$
$\tau^{\mathcal{U}}_{\mathcal{I}} := (com^*_{\mathcal{I}}, d^*, \sigma^*, s^*, u_1^*)$
$\mathfrak{Q}_{\mathcal{U}} := \emptyset$
for $(\mathsf{pk}_{\mathcal{U}}{}^*, qid) \in \mathfrak{R}$ where $\mathsf{pk}_{\mathcal{U}}{}^* \stackrel{?}{=} \mathsf{pk}_{\mathcal{U}}$ **do**
  $s^*, u_1^* \leftarrow \mathbb{Z}_p$
  $d^* := C_{\mathcal{SP}}.\mathtt{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{SP}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{SP}}}, (s^*, ctr^{\mathcal{U}}_{qid}, \mathsf{sk}_{\mathcal{U}}, u_1^*))$
  $\tau^{\mathcal{U},qid}_{\mathcal{SP}} := (com^*_{\mathcal{SP},qid}, d^*, \sigma^*, s^*, u_1^*, \mathsf{regMN}^{\mathcal{U}}_{qid}, ctr^{\mathcal{U}}_{qid})$
  $\mathfrak{Q}_{\mathcal{U}} := \mathfrak{Q}_{\mathcal{U}} \cup \{qid, \tau^{\mathcal{U},qid}_{\mathcal{SP}}\}$
endfor
$\mathfrak{S}_{\mathcal{U}} := \emptyset$
for $\tau'_{\mathcal{V}} \in \mathfrak{S}'_{\mathcal{U}}$ **do**
  $(com_{\mathcal{V}}, r', \mathsf{k}_{\mathrm{mac}}) := \tau'_{\mathcal{V}}$
  $d_{\mathcal{V}} := C_{\mathcal{V}}.\mathtt{Equiv}(\mathsf{CRS}_{\mathsf{com}_{\mathcal{V}}}, \mathsf{td}_{\mathsf{com}_{\mathcal{V}}}, \mathsf{sk}_{\mathcal{U}})$
  $\tau_{\mathcal{V}} := (com_{\mathcal{V}}, d_{\mathcal{V}}, \mathsf{k}_{\mathrm{mac}})$
  $\mathfrak{S}_{\mathcal{U}} := \mathfrak{S}_{\mathcal{U}} \cup \{\tau_{\mathcal{V}}\}$
endfor
$\mathfrak{S}_{\mathcal{U}} := \mathfrak{S}_{\mathcal{U}} \cup \mathfrak{S}^*_{\mathcal{U}}$
return $(\tau^{\mathcal{U}}_{\mathcal{I}}, \mathfrak{Q}_{\mathcal{U}}, \mathfrak{S}_{\mathcal{U}})$

Figure 7.54: Sim algorithm used in $\mathrm{Corrupt}_7$

Note that with Game 7, we finally reached $\text{Exp}^{\text{TU-ideal}}_{\text{GI},\mathcal{A}}$. As all the previous games have been indistinguishable from its predecessors, we have shown that Game 7 is indistinguishable from Game 1 and, thus, $\text{Exp}^{\text{TU-real}}_{\text{GI},\mathcal{A}}$ is indistinguishable from $\text{Exp}^{\text{TU-ideal}}_{\text{GI},\mathcal{A}}$, which concludes our proof.

7.6.10 False accusation protection

Theorem 7.25 (False accusation protection) *If* GI *provides* transaction unlinkability *and the* CDH *assumption holds with respect to* G_1*, then* GI *provides* false claim protection.

Proof idea The proof idea of the corresponding BBA+ property still holds. Assume there is an efficient adversary $\mathcal{A}$ breaking the *false accusation protection* of GI. This adversary has access to the oracles RealHonIssue, RealHonRegisterUser, RealHonSubmitReport RealHonCollect and RealHonRedeem for a user identity $\mathsf{pk}_{\mathcal{U}}$ created by the experiment. Note that this is a strict subset of the oracles from the *transaction unlinkability* experiment. We replace all this oracles by the ideal world oracles SimHonIssue, SimHonRegisterUser, SimHonCollect and SimHonRedeem and modify the setup algorithm to allow the simulation of proofs and equivocation of commitments (cf. Figure 7.48). We distinguish between two cases:

1. The proof Π that $\mathcal{A}$ outputs is still valid with non-negligible probability. As $\Pi = \mathsf{sk}_{\mathcal{U}}$, this essentially means that $\mathcal{A}$ can compute the discrete logarithm of $\mathsf{pk}_{\mathcal{U}}$, as all the simulation oracles are PPT and $\mathsf{pk}_{\mathcal{U}}$ is the only input they get that is not independent of $\mathsf{sk}_{\mathcal{U}}$. Therefore, in this case, we could use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ against the CDH assumption in G_1. $\mathcal{A}'$ simulates the experiment and the user oracles for $\mathcal{A}$. Let $g^x = X, g^y = Y$ be the challenge G_1 elements send by the adversary. The $\mathcal{A}'$ replaces $\mathsf{pk}_{\mathcal{U}}$ with X. Note that for the simulation of the oracles, no knowledge of $\mathsf{sk}_{\mathcal{U}} = x$ is required. When $\mathcal{A}$ outputs a valid proof $\Pi = \mathsf{sk}_{\mathcal{U}} = x$, $\mathcal{A}'$ outputs $Y^{\Pi} = (g^y)^x$ and wins the CDH experiment.

2. $\mathcal{A}$ cannot output a valid proof Π with more than negligible probability any longer. In this case, we could use $\mathcal{A}$ to construct an adversary $\mathcal{A}'$ against the *transaction unlinkability* property of GI. First $\mathcal{A}'$ generates an honest user for $\mathcal{A}$ by calling its HonUser oracle. Then $\mathcal{A}'$ forwards all the oracle calls from $\mathcal{A}$ to its own oracles. After $\mathcal{A}$ outputs Π and terminates, $\mathcal{A}'$ verifies Π using `VerifyGuilt` and outputs the result as its decision whether it interacts with the real or ideal world experiments. As Π is valid with non-negligible probability in the real world experiment and only with negligible probability in the ideal world experiment, $\mathcal{A}'$ can distinguish between those two experiments with non-negligible probability.

7.7 Performance of the instantiation

We focus on the performance of the code executed by the users as they are the most resource-constrained entities in our system. We assume that the user code will be executed on smartphones, which are powerful

compared to the embedded devices used in traditional WSNs. As our protocols are similar to BBA+, we use their profiling results as a baseline to estimate the performance of our system. We then estimate the performance impact of our modifications.

For the evaluation of BBA+, Hartung et al. [Har+19] made use of a OnePlus 3 smartphone with a Snapdragon 820 Quad-Core Processor (2×2.15 GHz & 2×1.6 GHz) and 6 GB RAM running Android v8.0.0. They used the open-source library RELIC [AG16] v0.4.1 with Barreto-Naehrig curves Fp254BNb and Fp254n2BNb and the optimal Ate pairing for the bilinear group. Executing the `Issue` protocol takes the user 117.92 ms without and 123.96 ms with optimizations. The `Accum` protocol takes 414.87 ms and 338.98 ms and the `Verify` protocol 399.93 ms and 329.73 ms with and without optimizations, respectively. Note that around 70%, in case of the optimized version, and around 80%, in case of the non-optimized version, of this time is spend on calculations that can be pre-computed before the transaction is initiated.

Herold et al. [Her+17] provides benchmarks for the computation of exponentiations in G_1 and G_2 and pairing evaluations for the same library, elliptic curves, and platform (except that they are using an older version of Android). We use estimation techniques from Klooß [Klo17] which give us an upper bound on the operations executed during Groth-Sahai proofs. Regarding proof verification, these estimations only apply if batch verification (cf. [Her+17]) is used as an optimization technique.

Using these techniques, we estimate that the `Issue` protocol of BBA+ uses 27 G_1 exponentiations, 18 G_2 exponentiations and 12 pairings. Combined with the benchmarks from [Her+17], we would obtain a total computation time of 116.61 ms, which is close to the real measurement results. Therefore, we conclude that we can combine those two results to estimate the performance of our system.

To calculate the proof size of the Groth-Sahai proofs, we use the numbers from [EG14].

As our `Issue` and `Redeem` protocols are identical to the `Issue` and `Verify` protocols of BBA+, we only examine the `RegisterUser`, `SubmitReport` and `Collect` protocols. We analyze the performance of our system without optimizations applied. Our results show that our system achieves practical performance on modern smartphones.

7.7.1 RegisterUser

The `RegisterUser` protocol is very similar to the `Issue` protocol from BBA+. The following differences exist:

- We require one additional G_1 exponentiation to compute the statement for the Groth-Sahai proof.
- We require one additional pairing evaluation to verify the signature in $\mathtt{UVer}_{\mathcal{SP}}$.
- We additionally have to verify the VRF proof.

Using the benchmarks from [Her+17], we would expect an additional execution time of 6.21 ms for the pairing and the G_1 exponentiation. We estimate that the Groth-Sahai proof of the VRF takes at most 263.04 ms using batch verification, yielding an estimated total execution time of 387.17 ms.

Regarding communication complexity, 2 $\mathbb{Z}_p$ elements, 25 G_1 elements and 22 G_2 elements have to be exchanged during the protocol run.

7.7.2 SubmitReport

The SubmitReport protocol compares to the Accum protocol of BBA+. Compared to Accum, the following modifications have been made:

- Additional generation of a MAC key with $M.\mathsf{Gen}(1^n)$. This only has negligible impact on the performance.
- Additional computation of a PEPSICo report with PI.ReportData.
- Additional computation of a commitment to the user's secret key $com_{\mathcal{V}}$ containing one $\mathbb{Z}_p$ element. This requires one additional G_1 exponentiation and two additional G_2 exponentiations.
- There is an additional G_1 element in the witness of the Groth-Sahai proof. This requires 4 additional G_1 exponentiations to compute the Groth-Sahai commitment.
- There is an additional equation in the Groth-Sahai proof to verify that the bulletin commitment $com_{\mathcal{V}}$ can be opened to the same $\mathsf{pk}_{\mathcal{U}}$ as the commitment com from the report counter token. This requires 8 additional G_1 exponentiations and 8 additional G_2 exponentiations.
- In the Groth-Sahai proof, we additionally have to verify that $ctr \neq 0$, ie. $g_1^{ctr} \neq 1$. Therefore, we require 5 additional G_1 exponentiations and 16 additional G_2 exponentiations.
- We require one additional pairing evaluation to verify the signature in $\mathsf{UVer}_{\mathcal{SP}}$.

Therefore, without the computation of the PEPSICo report, we would expect SubmitReport to take an additional 63.6 ms compared to Accum. Computing the PEPSICo report is an encryption of the IBE scheme. We suggest the usage of the Boneh-Franklin IBE, which uses symmetric bilinear groups for which we do not have benchmarks on the same platform. However, the performance is almost identical to an ElGamal encryption, with one additional pairing operation, which, however, is independent of the message and, therefore, only has to be executed once. To support larger message sizes, hybrid encryption can be used.

During the protocol run, 4 $\mathbb{Z}_p$ elements, 49 G_1 elements and 61 G_2 elements have to be exchanged in addition to the IBE encryption.

7.7.3 Collect

We compare the Collect protocol to the Accum protocol of BBA+.

- There is an additional G_2 element in the statement.
- There is an additional G_1 element in the witness. This requires 4 additional G_1 exponentiations to compute the Groth-Sahai commitment.
- There is an additional equation in the Groth-Sahai proof to verify that the bulletin commitment $com_{\mathcal{V}}$ can be opened to the same $\mathsf{pk}_{\mathcal{U}}$ as the commitment com from the balance token. This requires 8 additional G_1 exponentiations and 8 additional G_2 exponentiations.
- There is an additional MAC computation, which only has negligible impact on the performance.

Those changes result in an additional 12 G_1 exponentiations, 8 G_2 exponentiations, which would take an additional 22.60 ms, resulting in an estimated total execution time of 437.47 ms.

During the protocol run, 6 $\mathbb{Z}_p$ elements, 46 G_1 elements, 54 G_2 elements and one bitstring $\{0,1\}^n$ has to be transmitted.

7.8 Discussion

While we have eliminated PEPSICo's restriction to an honest-but-curious RA regarding *transaction unlinkability* and achieved forward privacy by directly using implementation details instead of its *report unlinkability* property, other limitations of PEPSICo still apply. Especially, the *data-hiding* and *subscription-hiding* properties of our system do not hold against an adversary that has compromised the SP and an honest user registered for the query identities in question. This is implied by the fact that the mobile node registration value is fixed per query identity (cf. Section 4.3). In our model, we used this fact in some of our proofs. Especially *transaction unlinkability* relies on this fact as the signatures that attest the validity of report counter tokens are over the mobile node registration value as well as over the corresponding commitment. As the signatures are known to the adversary and exposed if a user is corrupted, the experiment would have to equivocate the signatures, which is not possible. However, using a VRF mapping to $\mathbb{Z}_p$, and thus resulting in mobile node registration values from $\mathbb{Z}_p$, one could include these values into the commitment. Then, the indistinguishability between real and random mobile node registration values could be reduced to the *transaction unlinkablity* of PEPSICo. However, depending on the implementation, forward privacy might not hold any longer. Note that in case of a collusion as mentioned above the data submitted in a report remains secret due to the security of the IBE scheme. How feasible the assumption that such a collusion does not occur is, depends on the verifications and restrictions for user registration.

In our system, a malicious user could collect incentives for a previously submitted report without sending the MAC tag verifying its delivery to the SP. Therefore, the SP would not be able to prove their successful delivery. We argue that there is no disadvantage for users in sending the tag and, thus, it is feasible to assume that the large majority of users will behave honestly in this regard. However, this could be further enhanced by using a fair exchange mechanism.

A reputation mechanism could be added by extending the balance token to include an additional reputation score. Such reputation scores are helpful for the queriers to evaluate the trustworthiness of the submitted data (cf. Section 2.4). However, there are some aspects to consider in the user of the reputation scores. Would they be exposed and attached to each report during report submission, this might allow the tracking of transactions based on the development of the transaction scores. Fortunately, BBA+ already provides the means to address this problem. Using range proofs, it would be possible for users to prove that their reputation exceeded a specific threshold, without exposing the actual score. For example, different reputation levels could be specified. Then, during `SubmitReport` the user would claim that he has a specific reputation level and proof this claim by providing a range proof that his reputation exceeds the threshold of the level. The reputation level could then be attached to the report

and the querier could use this information when determining the trustworthiness of the submitted data. Together with the posted incentives, the querier could specify an update to the users' reputation score within a specific range defined by the system.

Note that we did not consider external threats such as entities eavesdropping on the network. Therefore, all protocol messages should additionally be encrypted with an IND-CCA secure encryption scheme. In addition, similar to PEPSICo, we did not consider privacy towards the network operator or identification based on communication details such as IP addresses. If the network operator is not trusted or identifying data would be exposed during the communication, an anonymizing network can be used.

Additionally, we did not consider possible side-channel attacks that might reveal some information about submitted reports and query identities, for example via statistics that might be published by the queriers together with the result of their research.

Moreover, our model does not prevent the identification or linkage of users based on the data within a single report. Therefore, the software that is used by users to submit data report has to provide absolute transparency on which data is collected. Furthermore, there would be potential legal problems with collecting personally identifiable information for query identities that are not exclusively available to one querier as the purpose the collected data will be used for might not be clearly defined in advance as, for example, it is required by the purpose limitation principle of the European General Data Protection Regulation (GDPR) [Eur16].

8 Conclusion

In this work, we presented I3PS, a participatory sensing system with incentive mechanism that provides strong security and privacy properties based on weak trust assumptions. We first introduced an interim model by combining PEPSICo and BBA+ in a black-box manner. We used this model to identify several issues arising from such a combination, which we addressed in I3PS.

I3PS protects user privacy by ensuring that transactions are unlinkable, even if against a collusion of malicious system operators, queriers, and other mobile nodes. Moreover, this also holds for transactions following a corruption of the user, except for the next incentive submission for each query identity where the corresponding report counter token has been exposed during the corruption and the next report collection. It keeps the reported data confidential from the SP and, in addition, hides the query identity of submitted reports and querier subscriptions as long as no user or querier registered for the same query identity colludes with the SP. Moreover, its incentive mechanism provides strong security guarantees towards malicious users and its report limitation mechanism provides a flexible way to limit the extent of the double-reporting problem. This is additionally backed by the prevention of incentive point sharing among users. Furthermore, queriers can verify that incentive points have been correctly delivered to the corresponding participants, offering some protection against malicious SPs and ISPs.

Therefore, our model effectively addresses the privacy issues in participatory sensing without requiring strong trust assumptions as in previous work.

As future work, the proposed model could be extended with a reputation mechanism to support the queriers in assessing data trustworthiness. Moreover, a reference implementation would help system operators to deploy the model in practical applications and would allow conducting precise performance benchmarks.

To further improve our model, a PEPSICo instantiation providing *node privacy* and *query privacy* against users registered for the challenge query identities could be helpful. However, it would remain to be seen how such an implementation would affect the other properties of our model. Furthermore, including a fair exchange mechanism in the incentive collection protocol, we could achieve *verifiable delivery* against malicious users.

Bibliography

[Abd+05] Michel Abdalla et al. "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions." In: *Annual International Cryptology Conference*. Springer. 2005, pp. 205–222.

[Abe+11] Masayuki Abe et al. "Optimal structure-preserving signatures in asymmetric bilinear groups." In: *Annual Cryptology Conference*. Springer. 2011, pp. 649–666.

[AG16] D. F. Aranha and C. P. L. Gouvêa. *RELIC is an Efficient LIbrary for Cryptography*. `https://github.com/relic-toolkit/relic`. 2016. URL: `https://github.com/relic-toolkit/relic` (visited on 12/14/2019).

[Bal+12] Jaime Ballesteros et al. "Safe cities. A participatory sensing approach." In: *37th Annual IEEE Conference on Local Computer Networks*. IEEE. 2012, pp. 626–634.

[BD93] Josh Benaloh and Michael De Mare. "One-way accumulators: A decentralized alternative to digital signatures." In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1993, pp. 274–285.

[Bel+09] Mira Belenkiy et al. "Compact e-cash and simulatable VRFs revisited." In: *International Conference on Pairing-Based Cryptography*. Springer. 2009, pp. 114–131.

[BF01] Dan Boneh and Matt Franklin. "Identity-based encryption from the Weil pairing." In: *Annual international cryptology conference*. Springer. 2001, pp. 213–229.

[BK13] I. Boutsis and V. Kalogeraki. "Privacy preservation for participatory sensing data." In: *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 2013, pp. 103–113.

[Bla+10] Olivier Blazy et al. "Batch groth–sahai." In: *International Conference on Applied Cryptography and Network Security*. Springer. 2010, pp. 218–235.

[Bur+06] Jeffrey A. Burke et al. "Participatory sensing." In: *ACM Sensys World Sensor Web Workshop* (2006).

[Cap+19] Andrea Capponi et al. "A Survey on Mobile Crowdsensing Systems: Challenges, Solutions and Opportunities." In: *IEEE Communications Surveys & Tutorials* (2019).

[CDB18] Martin Connolly, Ivana Dusparic, and Mélanie Bouroche. "An Identity Privacy Preserving Incentivization Scheme for Participatory Sensing." In: *2018 Eleventh International Conference on Mobile Computing and Ubiquitous Network (ICMU)*. IEEE. 2018, pp. 1–6.

[Chr+11a] Delphine Christin et al. "A survey on privacy in mobile participatory sensing applications." In: *Journal of systems and software* 84.11 (2011), pp. 1928–1946.

[Chr+11b] Delphine Christin et al. "Privacy-preserving collaborative path hiding for participatory sensing applications." In: *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*. IEEE. 2011, pp. 341–350.

[Chr+13] Delphine Christin et al. "Incognisense: An anonymity-preserving reputation framework for participatory sensing applications." In: *Pervasive and mobile Computing* 9.3 (2013), pp. 353–371.

[Chr+14] Delphine Christin et al. "TrustMeter: A trust assessment scheme for collaborative privacy mechanisms in participatory sensing applications." In: *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. 2014.

[Chr15] Delphine Christin. "Privacy in mobile participatory sensing: Current trends and future challenges." In: *Journal of Systems and Software* 116 (2016) (2015), pp. 57–68.

[Con+19] Martin Connolly et al. "Privacy Aware Incentivization for Participatory Sensing." In: *Sensors* 19.4049 (2019).

[Cot+08] Lance M. Cottrell et al. *Mixmaster*. 2008. URL: `http://mixmaster.sourceforge.net/` (visited on 12/14/2019).

[De +13] Yves-Alexandre De Montjoye et al. "Unique in the crowd: The privacy bounds of human mobility." In: *Scientific reports* 3 (2013), p. 1376.

[DHS15] David Derler, Christian Hanser, and Daniel Slamanig. "Revisiting cryptographic accumulators, additional properties and relations to other primitives." In: *Cryptographers' Track at the RSA Conference*. Springer. 2015, pp. 127–144.

[DKS12] Tassos Dimitriou, Ioannis Krontiris, and Ahmad Sabouri. "PEPPeR: A Querier's Privacy Enhancing Protocol for PaRticipatory Sensing." In: *Security and Privacy in Mobile Information and Communication Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 93–106.

[DS11] Emiliano De Cristofaro and Claudio Soriente. "Short Paper: PEPSI — privacy-enhanced Participatory Sensing Infrastructure." In: *Proceedings of the Fourth ACM Conference on Wireless Network Security*. WiSec '11. Hamburg, Germany: ACM, 2011, pp. 23–28.

[DS13] Emiliano De Cristofaro and Claudio Soriente. "Extended capabilities for a privacy-enhanced participatory sensing infrastructure (PEPSI)." In: *IEEE Transactions on Information Forensics and Security* 8.12 (2013), pp. 2021–2033.

[EG14] Alex Escala and Jens Groth. "Fine-Tuning Groth-Sahai Proofs." In: *Proceedings of the 17th International Conference on Public-Key Cryptography — PKC 2014 - Volume 8383*. New York, NY, USA: Springer-Verlag New York, Inc., 2014, pp. 630–649.

[ElG84] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms." In: *Advances in Cryptology*. Springer Berlin Heidelberg, 1984, pp. 10–18.

[Eur16] European Parliament and Council. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46 (General Data Protection Regulation)." In: *Official Journal of the European Union (OJ)* 59.1-88 (2016), p. 294.

[Gao+12] Sheng Gao et al. "Towards Location and Trajectory Privacy Protection in Participatory Sensing." In: *Mobile Computing, Applications, and Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 381–386.

[Gao+15] Hui Gao et al. "A survey of incentive mechanisms for participatory sensing." In: *IEEE Communications Surveys & Tutorials* 17.2 (2015), pp. 918–943.

[GGP14] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. "SPPEAR: security & privacy-preserving architecture for participatory-sensing applications." In: *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*. ACM. 2014, pp. 39–50.

[GGP16] Stylianos Gisdakis, Thanassis Giannetsos, and Panagiotis Papadimitratos. "Security, privacy, and incentive provision for mobile crowd sensing systems." In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 839–853.

[GHH15] Lin Gao, Fen Hou, and Jianwei Huang. "Providing long-term participation incentive in participatory sensing." In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE. 2015, pp. 2803–2811.

[GMP14] Felix Günther, Mark Manulis, and Andreas Peter. "Privacy-enhanced participatory sensing with collusion resistance and data aggregation." In: *International Conference on Cryptology and Network Security*. Springer. 2014, pp. 321–336.

[Gol+09] Jeffrey Goldman et al. "Participatory Sensing: A citizen-powered approach to illuminating the patterns that shape our world." In: *Foresight & Governance Project, White Paper* (2009), pp. 1–15.

[GS08] Jens Groth and Amit Sahai. "Efficient non-interactive proof systems for bilinear groups." In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2008, pp. 415–432.

[Guo+14] B. Guo et al. "From participatory sensing to Mobile Crowd Sensing." In: *Proc. IEEE Int. Conf. Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*. Mar. 2014, pp. 593–598.

[Har+17] Gunnar Hartung et al. "BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection." In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2017, pp. 1925–1942.

[Har+19] Gunnar Hartung et al. *BBA+: Improving the Security and Applicability of Privacy-Preserving Point Collection – Full Version*. 2019.

[Has+12] David Hasenfratz et al. "Participatory air pollution monitoring using smartphones." In: *Mobile Sensing* 1 (2012), pp. 1–5.

[HCG15] Daojing He, Sammy Chan, and Mohsen Guizani. "User privacy and data trustworthiness in mobile crowd sensing." In: *IEEE Wireless Communications* 22.1 (2015), pp. 28–34.

[Her+17] Gottfried Herold et al. "New Techniques for Structural Batch Verification in Bilinear Groups with Applications to Groth-Sahai Proofs." In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2017, pp. 1547–1564.

[HHT18] Alex Haig, Shirley Anugrah Hayati, and Anthony Tomasic. "Real-time detection of crowded buses via mobile phones." In: (2018).

[HJ16] Dennis Hofheinz and Tibor Jager. "Verifiable random functions from standard assumptions." In: *Theory of Cryptography Conference*. Springer. 2016, pp. 336–362.

[HKH12] Kuan Lun Huang, Salil S. Kanhere, and Wen Hu. "A privacy-preserving reputation system for participatory sensing." In: *37th Annual IEEE Conference on Local Computer Networks*. 2012, pp. 10–18.

[Jin+16] Haiming Jin et al. "Enabling Privacy-Preserving Incentives for Mobile Crowd Sensing Systems." In: *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. 2016, pp. 344–353.

[JR16] Tibor Jager and Andy Rupp. "Black-Box Accumulation: Collecting Incentives in a Privacy-Preserving Way." In: *Proceedings on Privacy Enhancing Technologies 2016*. 2016.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. 2007.

[Klo+19] Lorenz Cuno Klopfenstein et al. ""Worth one minute": An anonymous rewarding platform for crowd-sensing systems." In: *Journal of Communications and Networks* 21.5 (2019), pp. 509–520.

[Klo17] Michael Klooß. "On the Efficiency of Non-Interactive Zero-Knowledge Proofs." Unpublished master's thesis. Karlsruhe Institute of Technology (KIT), 2017.

[Kru07] John Krumm. "Inference Attacks on Location Tracks." In: *Pervasive Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 127–143.

[KS11] Leyla Kazemi and Cyrus Shahabi. "A privacy-aware framework for participatory sensing." In: *ACM SIGKDD Explorations Newsletter* 13.1 (2011), pp. 43–51.

[KS12] Leyla Kazemi and Cyrus Shahabi. "TAPAS: Trustworthy privacy-aware participatory sensing." In: *Knowledge and Information Systems* 37.1 (2012), pp. 105–128.

[LC12] Qinghua Li and Guohong Cao. "Efficient and privacy-preserving data aggregation in mobile sensing." In: *2012 20th IEEE International Conference on Network Protocols (ICNP)*. 2012.

[LC13] Qinghua Li and Guohong Cao. "Efficient Privacy-Preserving Stream Aggregation in Mobile Sensing with Low Aggregation Error." In: *Privacy Enhancing Technologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 60–81.

[MK14] Antonis Michalas and Nikos Komninos. "The lord of the sense: A privacy preserving reputation system for participatory sensing applications." In: *2014 IEEE Symposium on Computers and Communications (ISCC)*. 2014.

[NI97] Julio C. Navas and Tomasz Imielinski. "GeoCast—geographic addressing and routing." In: *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*. Citeseer. 1997, pp. 66–76.

[Sch+11] Immanuel Schweizer et al. "NoiseMap-real-time participatory noise maps." In: *Second international workshop on sensing applications on mobile phones*. Citeseer. 2011, pp. 1–5.

[Sha84] Adi Shamir. "Identity-based cryptosystems and signature schemes." In: *Workshop on the theory and application of cryptographic techniques*. Springer. 1984, pp. 47–53.

[Shi+11] Minho Shin et al. "AnonySense: A system for anonymous opportunistic sensing." In: *Pervasive and Mobile Computing* 7.1 (2011), pp. 16–30.

[SS98] P. Samarati and L. Sweeney. *Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression*. Tech. rep. 1998.

[TGS14] Hien To, Gabriel Ghinita, and Cyrus Shahabi. "A Framework for Protecting Worker Location Privacy in Spatial Crowdsourcing." In: *Proceedings of the VLDB Endowment* 7.10 (2014), pp. 919–930.

[Til13] Sameer Tilak. "Real-world deployments of participatory sensing applications: Current trends and future directions." In: *ISRN Sensor Networks* 2013 (2013).

[Tor19] The Tor Project, Inc. *The Tor Project | Privacy & Freedom Online*. 2019. URL: `https://www.torproject.org` (visited on 12/14/2019).

[VZG12] Khuong Vu, Rong Zheng, and Jie Gao. "Efficient algorithms for k-anonymous location privacy in participatory sensing." In: *2012 Proceedings IEEE International Conference on Computer Communications (INFOCOM)*. IEEE. 2012, pp. 2399–2407.

[WAK15] Dong Wang, Tarek Abdelzaher, and Lance Kaplan. *Social sensing: building reliable systems on unreliable data*. Morgan Kaufmann, 2015.

[Wan+19] Yongfeng Wang et al. "Privacy protection in mobile crowd sensing: a survey." In: *World Wide Web* (2019). Springer Science and Business Media LLC.

[Zha+12] Fan Zhang et al. "Data perturbation with state-dependent noise for participatory sensing." In: *2012 Proceedings IEEE International Conference on Computer Communications (INFOCOM)*. 2012, pp. 2246–2254.

[ZZL12] Pengfei Zhou, Yuanqing Zheng, and Mo Li. "How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing." In: *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM. 2012, pp. 379–392.

Figures

Definitions

Theorems and lemmas

Acronyms

APC Automated Passenger Counting. 5

BBA Black-box Accumulation. 35

BBA+ Extended Black-box Accumulation. v, vii, 1, 2, 11, 19, 35–46, 51, 52, 55, 59, 60, 64–68, 70, 76, 85, 90, 99, 103, 106, 107, 111, 113, 117, 127, 128, 133, 136, 146–149, 151, 159, 169

CRS Common Reference String. 16–19, 24, 25, 35, 38, 70, 71, 83, 100, 101, 104, 105, 125, 136, 137

GDPR European General Data Protection Regulation. 150

GPS Global Positioning System. 5

I3PS Incentivized Privacy-Preserving Participatory Sensing. 1, 48, 50, 55, 71, 74, 79, 80, 82, 83, 87–89, 99, 119, 125, 126, 151, 160, 161, 163, 164

IBE Identity-based Encryption. 21, 22, 69, 101, 136, 148, 149, 159, 163, 169

ID identifier. 10, 37, 39, 77, 84, 101, 104, 106, 124, 126, 131, 132

IP Internet Protocol. 6, 9, 150

IPPI Identity Privacy Preserving Incentivization. 9

ISP Incentive System Provider. 11, 46–52, 65, 66, 70–74, 76, 77, 84, 85, 88, 90, 92, 99, 103–106, 111, 113, 117, 129, 134, 151

MAC Message Authentication Code. 13, 14, 69, 73, 100, 101, 104, 117, 135, 148, 149, 159, 163

MCS Mobile Crowd Sensing. 4

MN Mobile Node. 27–29, 32, 46, 47

NIZK Non-interactive Zero-Knowledge proof system. 24, 100, 104, 111, 113, 127, 128, 163

NO Network Operator. 34

PAI Privacy-Aware Incentivization. 9

PEPSI Privacy-enhanced Participatory Sensing Infrastructure. 27, 34

PEPSICo Privacy-enhanced Participatory Sensing Infrastructure with Collusion Resistance. v, vii, 1–3, 6, 7, 21, 27–34, 45, 46, 48, 51, 55, 57, 59, 62, 63, 65, 67, 69, 70, 76, 78, 83, 90, 99, 101, 102, 104, 107, 111, 117, 119, 125, 126, 148–151, 159–161, 163

PII Personally Identifying Information. 1, 6

PKE Public Key Encryption. 19–21, 101, 159, 163

PRF Pseudo Random Function. 12, 16, 69, 101

RA Registration Authority. 27–29, 32, 33, 46–49, 59, 62, 68–72, 83, 85, 90, 91, 99, 101, 103, 105, 107, 111, 113, 119, 120, 123, 125, 129, 135, 149, 169

SP Service Provider. 27–29, 31–34, 46–49, 52, 65, 68–73, 76, 78, 83, 88, 90, 99, 103–105, 111, 113, 117, 119, 120, 122–124, 126, 129, 130, 134, 149, 151, 169

TTP Trusted Third Party. 21, 35, 36, 38, 46–48, 70–72, 83

VRF Verifiable Random Function. 16, 17, 69, 101, 102, 136, 139, 147, 149, 159, 160, 163

WOM ”Worth One Minute”. 8

WSN Wireless Sensor Network. 4, 5, 147

A One-time token mechanism

As an alternative to the report counter mechanism, one-time tokens can be used to limit the number of reports that a user can submit for a specific query identity. For the one-time token mechanism, we replace the `RegisterMN` and the `ReportData` algorithms from the interim model with the interactive protocols `RegisterUser` and `SubmitReport` respectively. We define them as follows.

$((\mathsf{regMN}_{qid}, \vec{t}, b_{\mathcal{U}}), b_{\mathcal{RA}}) \leftarrow \texttt{RegisterUser} \left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \right\rangle$

When registering for a query identity, the RA has to check the identity of the user and that the user is not already registered. Therefore, the protocol is identifying. In addition to the registration value regMN_{qid}, the user obtains max_{qid} one-time tokens $\vec{t}$.

$(b_{\mathcal{U}}, (c, com, b_{\mathcal{I}})) \leftarrow \texttt{SubmitReport} \left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \mathsf{regMN}_{qid}, qid, m, t), \mathcal{SP}(\mathsf{pk}_{\mathcal{RA}}) \right\rangle$

With each data report, the user has to send the commitment com and signature σ from a one-time token t, which is verified by the SP for validity and freshness. Moreover, the SP has to verify that the submitted report c, which has to be sent to the SP during the protocol, contains the same mobile node registration value as the one-time tokens are issued for.

A possible instantiation of these protocols is given in Figure A.1 and Figure A.2, respectively. $(G_1, G_2, G_T, e, p, g_1, g_2) \leftarrow \texttt{SetupGrp}(1^n)$ is the description of a bilinear group where the SXDH problem is assumed to be hard. We use a Commitment scheme C and Signature scheme S and IBE scheme E compatible with the Groth-Sahai based implementation of BBA+ given in [Har+17]. $P1$ and $P2$ are non-interactive Zero-Knowledge proof systems for the languages L_1 and L_2, respectively. They are defined as beneath the protocols. Furthermore, $qid \in \mathbb{Z}_p$ and $\mathsf{regMN}_{qid} \in G_1$.

$\texttt{RegisterUser}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid), \mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid}) \right\rangle$

$\mathcal{U}(\mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, qid)$		$\mathcal{RA}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{U}}, qid, max_{qid})$
	$\xleftarrow{\mathsf{regMN}_{qid}, max_{qid}}$	$\mathsf{regMN}_{qid} \leftarrow \texttt{RegisterMN}_{\mathrm{PI}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{sk}_{\mathcal{RA}}, qid)$
for $i = 0$ to $(max_{qid} - 1)$ **do**		
$r_i \xleftarrow{\$} \mathbb{Z}_p$		
$(com_i, d_i) \leftarrow C.\mathsf{Com}(r_i, \mathsf{sk}_{\mathcal{U}})$		
$stm \leftarrow (com_i, \mathsf{pk}_{\mathcal{U}})$		
$wit \leftarrow (g_2^{\mathsf{sk}_{\mathcal{U}}}, g_1^{r_i}, d_i)$		
$\pi \leftarrow P1.\mathsf{Prove}(\mathsf{CRS}, stm, wit)$	$\xrightarrow{com_i, \pi}$	
		$stm \leftarrow (com_i, g_1^{qid}, \mathsf{pk}_{\mathcal{U}})$
		if $P1.\mathsf{Verify}(\mathsf{CRS}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return 0
		endif
	$\xleftarrow{\sigma_i}$	$\sigma_i \leftarrow S.\mathsf{Sign}(\mathsf{sk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, com_i)$
endfor		
$\vec{t} := ((com_0, d_0, r_0, \sigma_0),$		
$\ldots,$		
$(com_{(max_{qid}-1)}, d_{(max_{qid}-1)},$		
$t_{(max_{qid}-1)}, \sigma_{(max_{qid}-1)}))$		
return $(\mathsf{regMN}_{qid}, \vec{t}, 1)$		**return** 1

$$L_1 := \left\{ com, \mathsf{pk}_{\mathcal{U}} \;\middle|\; \begin{array}{l} \exists SKU \in G_2; \\ R, d \in G_1 : \\ C.\mathsf{Open}(\mathsf{CRS}, com, d, (R, \mathsf{pk}_{\mathcal{U}}) \stackrel{?}{=} 1 \\ e(\mathsf{pk}_{\mathcal{U}}, g_2) \stackrel{?}{=} e(g_1, SKU) \end{array} \right\}$$

Figure A.1: `RegisterUser` protocol for the one-time token mechanism

$\texttt{SubmitReport}\left\langle \mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \mathsf{regMN}_{qid}, qid, m, t), \mathcal{SP}(\mathsf{pk}_{\mathcal{RA}}) \right\rangle$

$\mathcal{U}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, \mathsf{regMN}_{qid}, qid, m, t)$		$\mathcal{SP}(\mathsf{pk}_{\mathcal{RA}})$
$r_c \leftarrow \mathbb{Z}_p$		
$c \leftarrow \texttt{ReportData}_{\mathrm{PI}}(\mathsf{pk}_{\mathcal{RA}}, \mathsf{regMN}_{qid}, qid, m)$		
$(com, d, r, \sigma) := t$		
$(com', d') \leftarrow C.Com(\mathsf{CRS}, (0,0,0))$		
$com'' := com \cdot com'$		
$d'' := d \cdot d'$		
$stm \leftarrow (com'', \mathsf{regMN}_{qid}, g_1^r, \mathsf{pk}_{\mathcal{RA}})$		
$wit \leftarrow (com, \sigma, d, d'', \mathsf{pk}_{\mathcal{U}}, \mathsf{sk}_{\mathcal{U}}, m)$		
$\pi \leftarrow P2.\texttt{Prove}(\mathsf{CRS}, stm, wit)$	$\xrightarrow{c, com, r, \pi}$	
		if $r \in \mathfrak{R}$ **then**
		return $(\perp, \perp, 0)$
		endif
		$\mathfrak{R} \leftarrow \mathfrak{R} \cup \{r\}$
		$(c_0, c_1) := c$
		$stm \leftarrow (com, c_0, g_1^r, \mathsf{pk}_{\mathcal{RA}})$
		if $P2.\texttt{Verify}(\mathsf{CRS}, stm, \pi) \stackrel{?}{=} 0$ **then**
		return $(\perp, \perp, 0)$
		endif
return 1		**return** $(c, com, 1)$

$$L_2 := \left\{ (com'', \mathsf{regMN}_{qid}, R, \mathsf{pk}_{\mathcal{RA}}) \;\middle|\; \begin{array}{l} \exists com \in G_2; \\ \sigma \in G_2^2 \times G_1; \\ d, d'', \mathsf{pk}_{\mathcal{U}} \in G_1; \\ \mathsf{sk}_{\mathcal{U}} \in \mathbb{Z}_p : \\ C.\texttt{Open}(\mathsf{CRS}, com, d, (R, \mathsf{pk}_{\mathcal{U}})) \stackrel{?}{=} 1 \\ C.\texttt{Open}(\mathsf{CRS}, com'', d'', (R, \mathsf{pk}_{\mathcal{U}})) \stackrel{?}{=} 1 \\ S.\texttt{Verify}(\mathsf{pk}_{\mathcal{RA}}, com, \mathsf{regMN}_{qid}, \sigma) \stackrel{?}{=} 1 \\ \mathsf{pk}_{\mathcal{U}} \stackrel{?}{=} g_1^{\mathsf{sk}_{\mathcal{U}}} \\ QID \stackrel{?}{=} g_1^{qid} \end{array} \right\}$$

Figure A.2: SubmitReport protocol for the one-time token mechanism