pRate: Anonymous Star Rating with Rating Secrecy

Jia Liu[1] and Mark Manulis[2]

[1] Thales UK Ltd, Cambridge, United Kingdom
jia.liu@thalesgroup.com
[2] Surrey Centre for Cyber Security, University of Surrey, United Kingdom
mark@manulis.eu

Abstract. We introduce pRate, a novel reputation management scheme with strong security and privacy guarantees for the users and their reputation scores. The reputation scores are computed based on the (aggregated) number(s) of stars that users receive from their raters. pRate allows users to advertise privacy-friendly statements about their reputation when searching for potential transaction partners. Ratings can only be submitted by partners who have been initially authorised by the ratee and issued a rating token. The scheme is managed by a possibly untrusted reputation manager who can register users and assist ratees in updating their reputation scores, yet without learning these scores. In addition to ensuring the secrecy of the ratings, a distinctive feature of pRate over prior proposals, is that it hides the identities of raters and ratees from each other during the transaction and rating stages. The scheme is built from a number of efficient cryptographic primitives; its security is formally modeled and proven to hold under widely used assumptions on bilinear groups.

1 Introduction

Establishing trust between prospective transaction partners on online platforms is a major challenge for today's digital economy. Reputation systems have gained popularity as an important risk assessment mechanism for measuring and managing the trustworthiness of involved parties and a variety of reputation systems is already deployed across many online marketplaces, e.g., eBay, Yelp!, BlaBlaCar, Airbnb, etc. Reputation is one of the most important assets of an individual and has its special market value [19, 34]. Online reviews are extremely influential for businesses. Studies have shown that 90% of customers read reviews before making a purchase decision and 94% of customers would use a business with a four-star rating [1].

In a reputation system, typically provided and managed by an online platform (reputation manager), each user is associated with a reputation score and transaction partners can rate each other, leave feedback and recommendations. The concern of repercussions thereby often deters users from providing honest ratings. It is particularly hard to get honest opinions when users are not anonymous, e.g., over 95% of Airbnb listings and almost all of the ratings of the BlaBlaCar rides are above 4.5-star scores which are overwhelmingly positive [34, 35]. The fear of retaliation is considered as a major factor for users to withhold truthful feedback after having some negative experience [31, 34]. Retaliation can be in any form such as unfairly low ratings, refusal of future transactions, or even physical assault. There is also fear that shared personal information can create racial and/or gender bias among users when choosing transaction partners. For example, Airbnb had to face racial discrimination complaints from African-American and Latino would-be renters. Moreover, the great amount of information shared on the online platform poses a serious threat to personal privacy and security. Take for instance car sharing or riding services where booked trips and travel patterns can be misused for person abductions or car thefts. The reputation score itself is also a potential threat to privacy as it can possibly be used to link users across transactions. The linkage between user activities and ratings can also de-anonymise users [28, 29].

Another point of concern is a potential bias from the service provider that manages the online reputation system. For many years, the online rating site Yelp! has been accused of removing positive reviews and highlighting negative ones, thus causing a massive downgrading of businesses in an attempt to force them to purchase advertisements [2–4, 19]. Hence protecting honest users and the integrity of their ratings from a potentially biased reputation manager is another important requirement in the design of the reputation systems. One way to achieve this is to prevent the reputation manager from learning the individual scores obtained by the users.

Various privacy-preserving reputation schemes [5, 6, 9–11, 15, 20–22] (see also Section 7 for more discussion) have been proposed to protect anonymity of raters and ratees to some extent, but none of these aims to hide reputation scores against the reputation manager. As far as we know, only [27, 30, 36] discuss the protection of reputation scores. The scheme in [27] hides rating scores from the users but not from the reputation manager and does not provide anonymity of the users. The scheme in [30] only supports unidirectional rating, i.e., a buyer can use pseudonyms to anonymously rate a service provider but not vice versa. Also, this system lacks accountability on users since users can arbitrarily create uncertified pseudonyms. AnonRep [36] is an anonymous reputation system designed for evaluating the quality of messages posted on a public board. Since any posted message can be rated by arbitrary users, AnonRep cannot be used to rate transactions where only transaction partners are allowed to rate each other.

Our contribution. We propose a star rating scheme called pRate that provides strong privacy and security guarantees for the users and their reputation scores. In our scheme, a reputation manager issues and updates reputation credentials for the users without learning the actual scores. Users can advertise their reputation scores anonymously and selectively to other parties. Two users are able to transact and rate each other without leaking any identity information to each other. Despite being anonymous, transaction partners are held accountable for their behaviours. This is achieved by enabling the reputation manager to learn the identities of transacting partners (but not the details of their transaction) when they submit their ratings, i.e., misbehaving users can thus be reported to and identified by the reputation manager. We also describe a batch accumulation mechanism which enables the reputation manager to aggregate multiple ratings, hiding the link between the transaction and the rating sessions, and preventing the ratee from learning individual rating values. Our construction is based on BBS+ signature [17], Bulletproofs [14], Chaum-Pedersen-Signed ElGamal Encryption [33], and several standard (non-interactive) zero-knowledge proofs of knowledge. We formally model and prove the security properties of pRate under well-known assumptions on bilinear maps in the random oracle model.

Organisation. The rest of this paper is organised as follows: Section 2 gives an overview of pRate. Section 3 presents formal definitions of its functionality and security. Section 4 describes the cryptographic assumptions and building blocks. The construction of pRate is specified in Section 5. Section 6 provides its security analysis. Section 7 describes other related work. The paper concludes in Section 8.

2 Overview of the pRate scheme

pRate is a star rating system that provides strong security guarantees for protecting reputation scores and user privacy. pRate is managed by the *reputation manager (RM)* who issues and updates reputation credentials for the users without learning their reputation scores. Users can advertise their reputation scores and exchange rating tokens enabling them to rate each other anonymously by submitting their (encrypted) ratings to the RM. A user i who submits a rating for another user j is called the *rater*, whereas user j is called the *ratee*.

In pRate each user's reputation is measured by a v-star score $(n_1, \cdots, n_v)$, where v is fixed and each n_i represents the number of received i-stars. For example, $v = 5$ implies that a single rating can contain up to five stars and a score $(10, 20, 30, 40, 50)$ in this case would mean that the user has received a total of ten 1-star, twenty 2-star, thirty 3-star, forty 4-star and fifty 5-star individual ratings. A user's reputation score is aggregated in a reputation credential which is blindly signed by the RM. Using the reputation credential, a user can publish an anonymous advertisement for their current reputation score. Instead of revealing the exact reputation score in the advertisement, the user can show that it satisfies some predicate P, e.g., that the average score is higher than four stars or that 90% of ratings are above three stars. Based on their advertisements, two anonymous users can establish an authenticated and confidential communication channel over which they exchange their rating tokens. These rating tokens encrypt the identities of the two transaction partners which can only be decrypted by the RM. Hence, both user identities are kept anonymous and unlinkable from each other during the transaction and rating phases. In order to rate each other, each transaction partner would use the received rating token to encrypt the rating value (which can only be decrypted by the ratee), sign and submit his encrypted rating to the RM. This phase does not require any online presence of the ratee and can be performed at any time. Each rating token has a unique serial number and can be used only once. Upon receiving the submitted (encrypted) rating, the RM extracts the identities of the rater and the ratee, aggregates the ratee's rating, and sends an update to the ratee enabling the latter to update his reputation credential. The rating tokens make users accountable for their behaviour and can also be used to directly report misbehaving users in case of wrongdoings. The fact that the RM is able to retrieve the identities of the rater and the ratee during the rating accumulation phase prevents a number of conventional attacks against reputation systems, such as Sybil attacks, self-promotion attacks, ballot stuffing attacks, whitewashing attacks and bad mouthing attacks [25]. Furthermore, pRate offers the following security and privacy properties:

– *Anonymity*: This property allows users to advertise their reputation scores and rate each other in an anonymous and unlinkable way. More specifically, a user can prove statements about his reputations score without revealing his identity and the actual score. Moreover, the link between the advertisements and ratings of transaction partners based on these advertisements remains unknown to the RM and any other users in the system. Multiple advertisements by the same user also remain unlinkable against other users and the RM. Different ratings submitted by the same user also remain unlinkable from the ratee's perspective.

- *Rating-secrecy*: Only the user knows the exact values from his reputation score. The submitted ratings are encrypted and thus remain hidden from the RM and other users. The RM can aggregate a newly submitted rating for some user into that user's reputation credential without learning the actual value of the rating.
- *Unforgeability*: This property ensures that none of the advertisements, rating tokens, or submitted rating values can be forged. In particular, any valid advertisement can only be generated by a user with a valid reputation credential which ensures that each user is accountable for their behaviours. It is impossible to forge a rating token exchanged during the transaction or to submit a forged rating for an honest user, even if the attacker corrupts the RM and other users. When a rater submits a new (encrypted) rating to the RM, the latter can check that the rater has been previously authorised by the ratee and that the rating is well-formed. A ratee can verify that updates for the reputation score received from the RM are correctly formed and were received from raters who have been previously authorised by the ratee. This ensures that the RM is not able to introduce fake ratings nor forge or modify ratings received from authorised honest raters.

In pRate, users publish anonymous advertisements to remain unlinkable across multiple transactions and use temporary public keys to set up independent secure communication channels for each transaction. Although the use of temporary keys preserves users' anonymity during the transaction and rating submission, we note that in real-world applications the identities of transacting users can be leaked due to some side-channel information. For example, in Airbnb, guests would meet home owners in person. For these applications, we provide a batch accumulation mechanism which enables the RM to aggregate multiple ratings for the same user prior to sending its reputation score update. This accumulation can be used to break the link between a single transaction session and the submitted rating for that transaction. The ratee would thus learn only the aggregated rating value from multiple transactions.

In addition to the above security properties, pRate is highly non-interactive and efficient. pRate allows users to publish and verify the reputation advertisements published by other users without any interaction with other users or the RM. Submission of ratings to the RM does not require online presence of the ratee, while the update of a rating by the ratee does not require interaction with the rater. The scheme provides short reputation credentials and utilises a number of efficient cryptographic mechanisms, including short and computationally efficient zero-knowledge proofs.

3 Syntax and security properties of pRate

In this section, we formalise the syntax of our pRate scheme and define its main security and privacy properties.

Definition 1. *A pRate scheme consists of the following polynomial-time algorithms and protocols:*

- $(\mathsf{ik}, \mathsf{ok}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$: *With this algorithm the RM initialises the rating scheme. On input of a security parameter λ, it outputs a master issuing key* ik*, a master opening key* ok*, and a set of public parameters* pp.
- $((\mathsf{urep}[i], \mathsf{scr}[i]) \leftarrow \mathsf{Join}(\mathsf{pp}, i), \mathsf{reg}[i] \leftarrow \mathsf{Issue}(\mathsf{pp}, \mathsf{ik}, i))$: *This is a registration protocol between a new user i and the RM, modeled as a pair of interactive algorithms,* Join *executed by i, and* Issue *executed by the RM. Upon successful completion, the protocol outputs a* reputation credential $\mathsf{urep}[i]$ *and an initial* reputation score $\mathsf{scr}[i]$ *to the user, and a* registration record $\mathsf{reg}[i]$ *to the RM which is stored in the registration database.*
- $(\mathsf{aid}, \pi_{\mathsf{rep}}) \leftarrow \mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{scr}[i], m, P)$: *With this algorithm any registered user i can anonymously advertise its reputation. The algorithm takes some message m and a predicate P as an additional input and outputs a reputation advertisement consisting of an* advertisement identifier aid *and a* reputation proof π_{rep}. *We consider m as a placeholder for any additional information about the advertised transaction. Moreover, since the advertiser is anonymous, we assume that m includes information on how the advertiser can be securely contacted by prospective transaction partners, e.g., by including some temporary public key for establishing a secure channel. With predicate P as an input to the algorithm we enable advertisements proving statements about user's* $\mathsf{scr}[i]$*, i.e. $P(\mathsf{scr}[i]) = 1$, without disclosing the score.*
- $1/0 \leftarrow \mathsf{RepVer}(\mathsf{pp}, \mathsf{aid}, \pi_{\mathsf{rep}}, m, P)$: *This algorithm verifies the validity of a published reputation advertisement, in particular of the reputation proof π_{rep} in relation to the predicate P. It outputs 1 if π_{rep} is valid and 0 otherwise.*
- $((\mathsf{sn}_0, \mathsf{sn}_1, \mathsf{RT}_0, \mathsf{UT}_0) \leftarrow \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_0], \mathsf{aid}_0, \mathsf{aid}_1), (\mathsf{sn}_1, \mathsf{sn}_0, \mathsf{RT}_1, \mathsf{UT}_1) \leftarrow \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_1], \mathsf{aid}_1, \mathsf{aid}_0))$: *This is an interactive protocol for the exchange of rating tokens between two users i_0 and i_1. Upon successful execution, the interactive algorithm* $\mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_b], \mathsf{aid}_b, \mathsf{aid}_{1-b})$ *run by each user i_b ($b \in \{0, 1\}$) outputs unique*

serial numbers $\mathsf{sn}_b, \mathsf{sn}_{1-b}$, *a* rating token RT_b *that user* i_b *will use to rate* i_{1-b}, *and an* update token UT_b *that* i_b *will retain and use later to update its own reputation credential.*

- $\delta \leftarrow \mathsf{RateGen}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{RT}, x)$: *This algorithm enables a user* i *to generate a rating. On input of the public parameters* pp, *a reputation credential* $\mathsf{urep}[i]$, *a rating token* RT *received from the transaction partner, and a chosen rating value* $x \in [1, v]$, *it outputs a* rating δ. *This algorithm may fail and output* $\perp$ *if the process could not be completed successfully, e.g.,* $x \notin [1, v]$.
- $(i, j, \mathsf{aux})/\perp \leftarrow \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$: *Using this algorithm the RM accumulates received ratings into the reputation credential of the ratee. Upon successful execution, the algorithm outputs the extracted rater's identity* i, *ratee's identity* j, *and an update information* aux, *which the RM sends to the ratee* j. *As part of this algorithm, the RM may also update the record* $\mathsf{reg}[j]$ *of the ratee. This algorithm may fail and output* $\perp$ *if the accumulation process could not be completed successfully, e.g., if the rating* δ *submitted by the rater is invalid.*
- $1/0 \leftarrow \mathsf{Upd}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{scr}[j], \mathsf{UT}, \mathsf{aux})$: *With this algorithm a rated user* j *after receiving the update information* aux *from the RM and in possession of the matching update token* UT *can update its own reputation credential* $\mathsf{urep}[j]$ *and score* $\mathsf{scr}[j]$. *The algorithm outputs 1 if the update is successful and 0 otherwise.*

A secure pRate scheme must possess the anonymity, rating-secrecy and unforgeability properties that were introduced informally in Section 2. In Appendix A we formalise these properties using game-based security definitions which are based on security models for group signatures [8, 13]. For unforgeability, we model three different aspects: (i) *advertisement-unforgeability* to ensure that only users in possession of a valid reputation credential urep can create valid advertisements for their reputations scores, (ii) *ratee-unforgeability* to ensure that only users in possession of a valid reputation credential urep can issue rating tokens during the execution of the Token protocol that can then be used to produce ratings, (iii) *rater-unforgeability* to ensure that only users in possession of rating tokens issued to them by some other user can submit valid ratings for that user.

4 Cryptographic building blocks and assumptions

In the following we recall some well-known assumptions on bilinear maps and cryptographic building blocks used in our scheme.

Bilinear Maps. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative groups of prime order p. A function $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map if it satisfies the following three properties:

1. Bilinear: $\hat{\mathsf{e}}(g^a, h^b) = \hat{\mathsf{e}}(g, h)^{ab}$ for all $g \in \mathbb{G}_1, h \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$.
2. Non-degenerate: there exists $g \in \mathbb{G}_1, h \in \mathbb{G}_2$ such that $\hat{\mathsf{e}}(g, h) \neq 1$.
3. Computable: $\hat{\mathsf{e}}(g, h)$ is efficiently computable for all $g \in \mathbb{G}_1, h \in \mathbb{G}_2$.

Our scheme can be implemented using both Type 2 and Type 3 pairings [24], as long as the XDH and q-SDH assumptions described below are supported.

EXternal Diffie-Hellman (XDH) Assumption [16]. Given groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ associated with a bilinear pairing $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The XDH assumption holds if the Decision Diffie-Hellman (DDH) problem is hard in $\mathbb{G}_1$.

q-SDH Assumption [12]. The q-Strong Diffie-Hellman (SDH) assumption states that given two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with generators g_1 for $\mathbb{G}_1$ and g_2 for $\mathbb{G}_2$, for any PPT adversary $\mathcal{A}$, the following advantage is negligible in λ: $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{q\text{-}SDH}}(1^\lambda) = \mathsf{Pr}[\mathcal{A}(g_1, g_1^\gamma, \cdots, g_1^{\gamma^q}, g_2, g_2^\gamma) = (g_1^{\frac{1}{\gamma+x}}, x) : \gamma \xleftarrow{\$} \mathbb{Z}_p^*]$.

BBS+ Signature [7, 17]. The BBS+ signature allows a signer to issue and update a signature on a tuple of messages in a blind way, i.e., without learning the values of the messages. In pRate these techniques are used to construct reputation credentials. A user in possession of a BBS+ signature can selectively disclose some messages and produce zero-knowledge proofs for statements about other messages.

Zero-knowledge proofs. Following the notations in [18], we use $\mathsf{PoK}\{(x) : h = g^x\}$ to denote a non-interactive zero-knowledge proof of knowledge of x that satisfies $h = g^x$ and use $\mathsf{SoK}[m]\{(x) : h = g^x\}$ to refer to a signature of knowledge on m. A *range proof* is a special zero-knowledge proof which shows that a committed value lies within a certain interval. Recent Bulletproofs [14] that do not require a trusted setup are used in our scheme to produce privacy-preserving statements about reputation scores.A proof that some secret lies within an interval statement $v \in [0, 2^n - 1]$ requires only $2\lceil \log n \rceil + 4$ group elements and 5 elements in $\mathbb{Z}_p$. Bulletproofs support aggregation, i.e., k range proofs, possibly over different intervals, can be combined into a single proof with only $2 \log k$ additional group elements.

Chaum-Pedersen-Signed ElGamal Encryption (CPS-EG) [33] The CPS-EG scheme is a modified version of the Schnorr-Signed ElGamal encryption that achieves IND-CCA2-security in the random oracle model. pRate uses the techniques from CPS-EG in generation of rating tokens and ratings. Its IND-CCA2 security provides decryption oracle that is used in the proof of anonymity of pRate.

5 Our pRate scheme

5.1 Specifications of pRate algorithms and protocols

In the following we provide detailed specifications of the algorithms and protocols behind the proposed pRate scheme which allows users to advertise their reputation, to rate and be rated by other users in a privacy-friendly way. The scheme is managed by the reputation manager RM. The communication between a user and the RM is assumed to be over secure channels.

Initialisation of the scheme. The algorithm $\mathsf{Setup}(1^\lambda)$ executed by the RM performs the following steps. Choose $\gamma, \xi \xleftarrow{\$} \mathbb{Z}_p^*$, $g_0, g_1, \cdots, g_{v+3}, g, u \xleftarrow{\$} \mathbb{G}_1, w \xleftarrow{\$} \mathbb{G}_2$, compute $W = w^\gamma$ and $U = u^\xi$. Output the master issuing key $\mathsf{ik} = \gamma$, the master opening key $\mathsf{ok} = \xi$, and the public parameters $\mathsf{pp} = (g_0, g_1, \cdots, g_{v+3}, g, w, W, u, U)$.

Registration of new users. The interactive protocol $(\mathsf{Join}(\mathsf{pp}, i), \mathsf{Issue}(\mathsf{pp}, \mathsf{ik}, i))$ executed between a new user i and the RM is specified below.

– $\mathsf{Join}(\mathsf{pp}, i)$:
  - User i chooses randoms $k, s_1 \xleftarrow{\$} \mathbb{Z}_p^*$. Compute $K = g_{v+2}^k, S_1 = g_{v+3}^{s_1}$ and a proof $\pi_{\mathsf{id}} = \mathsf{PoK}\{(k, s_1) : K = g_{v+2}^k \wedge S_1 = g_{v+3}^{s_1}\}$ (see Figure 2 for details). User i sends $(K, S_1, \pi_{\mathsf{id}})$ to the RM.
  - Upon receiving $(n_1, \cdots, n_v, t, e, s_2, C)$ from RM, user i computes $s = s_1 + s_2$ and $R = g_0 g_1^{n_1} \cdots g_v^{n_v} g_{v+1}^t g_{v+2}^k g_{v+3}^s$. Verify if $\hat{\mathsf{e}}(C, W \cdot w^e) \stackrel{?}{=} \hat{\mathsf{e}}(R, w)$. If successful, set $\mathsf{urep}[i] = (k, s, e, R, C)$ and $\mathsf{scr}[i] = (n_1, n_2, \cdots, n_v, t)$.

– $\mathsf{Issue}(\mathsf{pp}, \mathsf{ik}, i)$: Upon receiving $(K, S_1, \pi_{\mathsf{id}})$ from user i, RM verifies if π_{id} is valid using the verification algorithm in Figure 2. If successful, select initial values $n_1, \cdots, n_v$ and a timestamp t. Choose $e, s_2 \xleftarrow{\$} \mathbb{Z}_p^*$. Compute $T = g_{v+1}^t, S_2 = g_{v+3}^{s_2}, R = g_0 g_1^{n_1} \cdots g_v^{n_v} T K S_1 S_2$ and $C = R^{\frac{1}{\gamma + e}}$. Set $\mathsf{reg}[i] = (K, e, t, R, C)$ and send $(n_1, \cdots, n_v, t, e, s_2, C)$ to user i.

We remark that the algorithm Join executed by the user i outputs the initial reputation credential $\mathsf{urep}[i] = (k, s, e, R, C)$ and score $\mathsf{scr}[i] = (n_1, n_2, \cdots, n_v, t)$ where the star values $n_1, \cdots, n_v$ can all be set to 0 or any other fixed values, which the system assigns to its new users. The timestamp t initially represents the time at which the reputation credential was issued. The secret key k stored in the reputation credential is the long-term key of the user i and is chosen by the user as part of the protocol. Its knowledge is proven in π_{id} along with the knowledge of randomness s_1. The randomness $s = s_1 + s_2$ which is used to seal the information stored in C is generated jointly by the user and the RM to ease the proof of advertisement-unforgeability. The algorithm Issue executed by the RM outputs the registration record $\mathsf{reg}[i] = (K, e, t, R, C)$ where $K = g_{v+2}^k$ represents the identity of the new user within the system. Some information from the user's reputation credential, i.e., (e, R, C), are stored by the RM and will be used later to compute reputation updates.

Reputation advertisements and their verification. The algorithm $\mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{scr}[i], m, P)$ outputs a reputation advertisement $(\mathsf{aid}, \pi_{\mathsf{rep}})$ by computing its identifier $\mathsf{aid} = (d, d^k)$ using some random $d \xleftarrow{\$} \mathbb{G}_1$, and the reputation proof $\pi_{\mathsf{rep}} = \mathsf{SoK}\,[m]\,\{(\mathsf{urep}[i], \mathsf{scr}[i]) : C = (g_0 g_1^{n_1} \cdots g_v^{n_v} g_{v+1}^t g_{v+2}^k g_{v+3}^s)^{\frac{1}{\gamma+e}} \wedge P(\mathsf{scr}[i]) = 1 \wedge \mathsf{aid} = (d, d^k)\}$ that shows the current user's score $\mathsf{scr}[i]$ satisfies some predicate P. In our specification P is left general to show support for arbitrary predicates with corresponding zero-knowledge proofs. Nonetheless, in Appendix B we show an example on how to create proofs for predicates $P = \bigwedge_{i=1}^{\zeta} \left(\psi_i \in [0, 2^{\ell_i})\right)$ involving interval statements computed from the number of stars $n_1, \cdots, n_v$ and timestamp t in the reputation score, using Bulletproofs [14], a recent zero-knowledge protocol for range proofs.

The algorithm $\mathsf{RepVer}(\mathsf{pp}, \mathsf{aid}, \pi_{\mathsf{rep}}, m, P)$ performs verification of the reputation proof π_{rep} and outputs 1 if the proof is valid, otherwise it outputs 0. Clearly, verification of π_{rep} involves verification of the zero-knowledge proof for $P(\mathsf{scr}[i]) = 1$ which depends on P (c.f. Appendix B for our example based on Bulletproofs [14]).

User i_0 : $\mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_0], \mathsf{aid}_0, \mathsf{aid}_1)$		User i_1 : $\mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_1], \mathsf{aid}_1, \mathsf{aid}_0))$
$a, r \xleftarrow{\$} \mathbb{Z}_p^*$,		$a', r' \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{ct}_1 = u^a, \mathsf{ct}_2 = g_{v+2}^{k_0} \cdot U^a, \mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$		$\mathsf{ct}'_1 = u^{a'}, \mathsf{ct}'_2 = g_{v+2}^{k_1} \cdot U^{a'}, \mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2)$
Create a proof $\pi_{\mathsf{enc}} = \mathsf{SoK}[r]\{(a, k_0) :$		Create a proof $\pi'_{\mathsf{enc}} = \mathsf{SoK}[r']\{(a', k_1) :$
$\mathsf{ct} = (u^a, g_{v+2}^{k_0} \cdot U^a) \wedge D_0 = d_0^{k_0}\}$		$\mathsf{ct}' = (u^{a'}, g_{v+2}^{k_1} \cdot U^{a'}) \wedge D_1 = d_1^{k_1}\}$
	$\xrightarrow{r, \mathsf{ct}, \pi_{\mathsf{enc}}}$	
	$\xleftarrow{r', \mathsf{ct}', \pi'_{\mathsf{enc}}}$	
If π'_{enc} is not valid then return $\perp$		If π_{enc} is not valid then return $\perp$
$\mathsf{sn} = g^r, \mathsf{sn}' = g^{r'}$		$\mathsf{sn} = g^r, \mathsf{sn}' = g^{r'}$
Create a proof $\pi_{\mathsf{tok}} = \mathsf{SoK}[\mathsf{sn}, \mathsf{sn}', \mathsf{ct}']$		Create a proof $\pi'_{\mathsf{tok}} = \mathsf{SoK}[\mathsf{sn}', \mathsf{sn}, \mathsf{ct}]$
$\{(a, k_0) : \mathsf{ct} = (u^a, g_{v+2}^{k_0} \cdot U^a)\}$		$\{(a', k_1) : \mathsf{ct}' = (u^{a'}, g_{v+2}^{k_1} \cdot U^{a'})\}$
	$\xrightarrow{\pi_{\mathsf{tok}}}$	
	$\xleftarrow{\pi'_{\mathsf{tok}}}$	
$\mathsf{RT} = (\mathsf{sn}, r, a, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}})$		$\mathsf{RT}' = (\mathsf{sn}', r', a', \mathsf{ct}', \mathsf{ct}, \mathsf{sn}, \pi_{\mathsf{tok}})$
$\mathsf{UT} = r'$		$\mathsf{UT}' = r$
Return $(\mathsf{sn}, \mathsf{sn}', \mathsf{RT}, \mathsf{UT})$		Return $(\mathsf{sn}', \mathsf{sn}, \mathsf{RT}', \mathsf{UT}')$

Fig. 1: Exchange of rating tokens. $\mathsf{urep}[i_b] = (k_b, s_b, e_b, R_b, C_b)$, $\mathsf{aid}_b = (d_b, D_b)$, $b \in \{0, 1\}$.

Exchange of rating tokens. The detailed specification of the Token protocol in which two prospective transaction partners i_0 and i_1 exchange their rating tokens is given in Figure 1 with the details of underlying zero-knowledge proofs π_{enc} and π_{tok} provided in Figure 2. It is assumed that both users have already obtained and verified their respective advertisements and have setup a secure channel prior to engaging in the Token protocol (cf. Section 5.2 for the discussion on anonymous advertisements and secure channels). We observe that the protocol is symmetric. User i_0 obtains the rating token RT and the update token UT whereas user i_1 obtains the rating-token RT' and the update-token UT'. Note that aid_b with $b = 0, 1$ are used to generate ciphertexts $\mathsf{ct}, \mathsf{ct}'$ that encrypt the identities K, K' of the users. Only RM can decrypt $\mathsf{ct}, \mathsf{ct}'$ with the opening key. The serial number $\mathsf{sn} = g^r$ resp. $\mathsf{sn}' = g^{r'}$ is used to ensure that each rating token can only be used once. The rating token computed by each user further includes randomness r resp. r' which corresponds to the update token retained by the other user.

Rating generation. A user in possession of a rating token received from another user can act as a rater for that user and prepare their own rating that will be submitted to the RM. The rating generation algorithm $\mathsf{RateGen}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{RT}, x)$ executed by user i with reputation credential $\mathsf{urep}[i] = (k, s, e, R, C)$ and the chosen rating value g_x with $x \in [1, v]$ performs the following steps. Parse $\mathsf{RT} = (\mathsf{sn}, r, a, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}})$. Verify π'_{tok} using the algorithm from Figure 2 to check whether RT is a valid rating token. Compute $V = g_x g_{v+3}^r$ and a proof $\pi_{\mathsf{sub}} = \mathsf{PoK}\{(r, a, x, k) : V = g_x g_{v+3}^r \wedge g_x \in \{g_1, \cdots, g_v\} \wedge \mathsf{sn} = g^r \wedge \mathsf{ct} = (u^a, g_{v+2}^k \cdot U^a)\}$ using the algorithm from Figure 2. Note that π_{sub} guarantees that V encrypts a valid rating $g_x \in \{g_1, \cdots, g_v\}$ and that user i is authorised to rate. Finally, output a rating $\delta = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$.

Rating accumulation. Upon receiving a new rating δ, the RM can check its validity, identify the rater and the ratee, and accumulate the new rating by issuing an update information to the ratee. The algorithm $\mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$ run by the RM proceeds as follows. Parse $\delta = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$ with $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and $\mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2)$. Check that the serial number sn has not been used before. Verify π'_{tok} and π_{sub} using the algorithms in Figure 2 to check the validity of δ. In the last step of verification of π_{tok} and π_{sub}, the opening key $\mathsf{ok} = \xi$ is used to compute $K = \mathsf{ct}_2/\mathsf{ct}_1^{\xi}$ and $K' = \mathsf{ct}'_2/{\mathsf{ct}'_1}^{\xi}$. Find registration records $\mathsf{reg}[i]$ and $\mathsf{reg}[j]$ such that $K_i = K$ and $K_j = K'$ to identify the user i who is rating user j. To accumulate δ into $\mathsf{reg}[j] = (K_j, e, t, R, C)$ with current time $\tilde{t}$, choose $\tilde{s} \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $\tilde{R} = R \cdot g_{v+1}^{\tilde{t}-t} \cdot V \cdot g_{v+3}^{\tilde{s}}$. Create a new reputation credential for user j by choosing a new random $\tilde{e} \xleftarrow{\$} \mathbb{Z}_p^*$ and computing $\tilde{C} = \tilde{R}^{\frac{1}{\gamma+\tilde{e}}}$. Update the registration record $\mathsf{reg}[j] \leftarrow (Z_j, \tilde{e}, \tilde{t}, \tilde{R}, \tilde{C})$, send $\mathsf{aux} = (\delta, \tilde{e}, \tilde{t}, \tilde{s}, \tilde{C})$ to user j, and output (i, j, aux).

Rating update. Upon receiving the update information from the RM users can update their own reputation credential and score. The algorithm $\mathsf{Upd}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{scr}[j], \mathsf{UT}, \mathsf{aux})$ executed by user j to update own reputation credential and score performs the following steps. Parse $\mathsf{urep}[j] = (k, s, e, R, C)$, $\mathsf{scr}[j] = (n_1, \cdots, n_v, t)$, $\mathsf{aux} = (\delta, \tilde{e}, \tilde{t}, \tilde{s}, \tilde{C})$ with $\delta = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$, and $\mathsf{UT} = r$. Verify validity of the update information aux by checking the proofs π'_{tok} and π_{sub}, checking $g^{\mathsf{UT}} \overset{?}{=} \mathsf{sn}$ and ensuring that UT hasn't been used in any previous update. If all successful, compute $g_x = V/g_{v+3}^{r}$ and $\tilde{R} = R \cdot g_{v+1}^{\tilde{t}-t} \cdot V \cdot g_{v+3}^{\tilde{s}}$. Check if $\hat{\mathsf{e}}(\tilde{C}, W \cdot w^{\tilde{e}}) \overset{?}{=} \hat{\mathsf{e}}(\tilde{R}, w)$. If successful, update $\mathsf{urep}[j] \leftarrow (k, s + r + \tilde{s}, \tilde{e}, \tilde{R}, \tilde{C})$ and $\mathsf{scr}[j] \leftarrow (n_1, \cdots, n_x + 1, \cdots, n_v, \tilde{t})$ and output 1, otherwise output 0.

$\pi_{\mathsf{id}} = \mathsf{PoK}\left\{(k, s_1) : K = g_{v+2}^{k} \wedge S_1 = g_{v+3}^{s_1}\right\}$

- $\mathsf{Prv}(k, s_1, g_{v+2}, g_{v+3}, K, S_1)$:
  - $r_k, r_s \xleftarrow{\$} \mathbb{Z}_p^*$
  - $R_1 = g_{v+2}^{r_k}, R_2 = g_{v+3}^{r_s}$
  - $c = \mathcal{H}(K, S_1, R_1, R_2)$
  - $\varrho_k = r_k + c \cdot k$, $\varrho_s = r_s + c \cdot s_1$
  - Return $\pi_{\mathsf{id}} = (c, \varrho_s, \varrho_k)$
- $\mathsf{Ver}(g_{v+2}, g_{v+3}, K, S_1, \pi_{\mathsf{id}})$:
  - Parse $\pi_{\mathsf{id}} = (c, \varrho_s, \varrho_k)$
  - $\hat{R}_1 = g_{v+2}^{\varrho_k}/K^{c}, \hat{R}_2 = g_{v+3}^{\varrho_s}/S_1^{c}$
  - $\hat{c} = \mathcal{H}(K, S_1, \hat{R}_1, \hat{R}_2)$
  - If $\hat{c} = c$ then return 1 else 0

$\pi_{\mathsf{enc}} = \mathsf{SoK}[r]\{(a, k) : \mathsf{ct} = (u^{a}, g_{v+2}^{k} \cdot U^{a}) \wedge D = d^{k}\}$

- $\mathsf{Prv}(a, k, u, g_{v+2}, U, \mathsf{ct}, d, D)$:
  - $r_a, r_k \xleftarrow{\$} \mathbb{Z}_p^*$
  - $R_1 = u^{r_a}, R_2 = g_{v+2}^{r_k} U^{r_a}, R_3 = d^{r_k}$
  - $c = \mathcal{H}(\mathsf{ct}, D, R_1, R_2, R_3)$
  - $\varrho_a = r_a + c \cdot a$, $\varrho_k = r_k + c \cdot k$
  - Return $\pi_{\mathsf{enc}} = (c, \varrho_a, \varrho_k)$
- $\mathsf{Ver}(u, g_{v+2}, U, \mathsf{ct}, d, D, \pi_{\mathsf{enc}})$:
  - Parse $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and $\pi_{\mathsf{enc}} = (c, \varrho_a, \varrho_k)$
  - $\hat{R}_1 = u^{\varrho_a}\mathsf{ct}_1^{-c}, \hat{R}_2 = g_{v+2}^{\varrho_k} U^{\varrho_a} \mathsf{ct}_2^{-c}, \hat{R}_3 = d^{\varrho_k} D^{-c}$
  - $\hat{c} = \mathcal{H}(\mathsf{ct}, D, \hat{R}_1, \hat{R}_2, \hat{R}_3)$
  - If $\hat{c} = c$ then return 1 else 0

$\pi_{\mathsf{tok}} = \mathsf{SoK}\,[\mathsf{sn}, \mathsf{sn}', \mathsf{ct}']\left\{(a, k) : \mathsf{ct} = (u^{a}, g_{v+2}^{k} \cdot U^{a})\right\}$

- $\mathsf{Prv}(a, u, U, \mathsf{ct}, \mathsf{sn}, \mathsf{sn}', \mathsf{ct}')$:
  - $r_a, r_k \xleftarrow{\$} \mathbb{Z}_p^*$, $R_1 = u^{r_a}, R_2 = U^{r_a}, R_3 = g_{v+2}^{r_k}, Z = U^{a}$
  - $c = \mathcal{H}(\mathsf{ct}, Z, R_1, R_2, R_3, \mathsf{sn}, \mathsf{sn}', \mathsf{ct}')$, $\varrho_a = r_a + c \cdot a$, $\varrho_k = r_k + c \cdot k$
  - Return $\pi_{\mathsf{tok}} = (\mathsf{ct}, c, \varrho_a, \varrho_k)$
- $\mathsf{Ver}(\xi, u, U, \pi_{\mathsf{tok}}, \mathsf{sn}, \mathsf{sn}', \mathsf{ct}')$:
  - Parse $\pi_{\mathsf{tok}} = (\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), c, \varrho_a, \varrho_k)$
  - $\hat{R}_1 = u^{\varrho_a}\mathsf{ct}_1^{-c}$, $\hat{R}_2 = \hat{R}_1^{\xi}$, $\hat{R}_3 = g_{v+2}^{\varrho_k} U^{\varrho_a} \hat{R}_2^{-1} \mathsf{ct}_2^{-c}$, $\hat{Z} = \mathsf{ct}_1^{\xi}$
  - $\hat{c} = \mathcal{H}(\mathsf{ct}, \hat{Z}, \hat{R}_1, \hat{R}_2, \hat{R}_3, \mathsf{sn}, \mathsf{sn}', \mathsf{ct}')$
  - If $c \neq \hat{c}$ or $U^{\varrho_a} \neq \hat{R}_2 \hat{Z}^{\hat{c}}$ return $\perp$
  - Return $m = \mathsf{ct}_2/\hat{Z}$

$\pi_{\mathsf{dec}} = \mathsf{PoK}\{(r') : H = h^{r'}\}$

- $\mathsf{Prv}(r')$:
  - $\theta \xleftarrow{\$} \mathbb{Z}_p^*$, $R = h^{\theta}$
  - $c = \mathcal{H}(H, R)$, $\varrho = \theta + c \cdot r'$
  - Return $\pi_{\mathsf{dec}} = (c, \varrho)$
- $\mathsf{Ver}(H, \pi_{\mathsf{dec}})$:
  - Parse $\pi_{\mathsf{dec}} = (c, \varrho)$
  - $\hat{R} = h^{\varrho}/H^{c}$
  - $\hat{c} = \mathcal{H}(H, \hat{R})$
  - If $\hat{c} = c$ then return 1 else 0

$\pi_{\mathsf{sub}} = \mathsf{PoK}\{(r, a, g_x, k) :\ V = g_x g_{v+3}^{r} \wedge g_x \in \{g_1, \cdots, g_v\} \wedge \mathsf{sn} = g^{r} \wedge \mathsf{ct} = (u^{a}, g_{v+2}^{k} \cdot U^{a})\}$

- $\mathsf{Prv}(r, a, g_x)$:
  - $r_a, r_k \xleftarrow{\$} \mathbb{Z}_p^*$, $R_1 = u^{r_a}, R_2 = U^{r_a}, R_3 = g_{v+2}^{r_k}, Z = U^{a}$
  - $\theta_x, \{e_j, \theta_j\}_{j=1, j\neq x}^{v} \xleftarrow{\$} \mathbb{Z}_p^*$
    - $A_x = g_{v+3}^{\theta_x}$, $B_x = g^{\theta_x}$
    - For $j \neq x$, $\varrho_j = \theta_j + e_j \cdot r$, $A_j = g_{v+3}^{\varrho_j}(g_j/V)^{e_j}$ and $B_j = g^{\theta_j}$
  - $c = \mathcal{H}(\mathsf{ct}, Z, R_1, R_2, R_3, A_1, \cdots, A_v, B_1, \cdots, B_v)$
  - $\varrho_a = r_a + c \cdot a$, $\varrho_k = r_k + c \cdot k$, $e_x = c - \sum_{j\neq x} e_j$, $\varrho_x = \theta_x + e_x \cdot r$
  - Return $\pi_{\mathsf{sub}} = (\mathsf{ct}, \varrho_a, \varrho_k, \{e_j, \varrho_j\}_{j=1}^{v})$
- $\mathsf{Ver}(\xi, V, \mathsf{sn}, \pi_{\mathsf{sub}})$:
  - Parse $\pi_{\mathsf{sub}} = (\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \varrho_a, \varrho_k, \{e_j, \varrho_j\}_{j=1}^{v})$
  - $c = \sum_{j=1}^{v} e_j$, $\hat{R}_1 = u^{\varrho_a}\mathsf{ct}_1^{-c}$, $\hat{R}_2 = \hat{R}_1^{\xi}$, $\hat{R}_3 = g_{v+2}^{\varrho_k} U^{\varrho_a} \hat{R}_2^{-1} \mathsf{ct}_2^{-c}$, $\hat{Z} = \mathsf{ct}_1^{\xi}$
  - $\hat{A}_j = g_{v+3}^{\varrho_j}(g_j/V)^{e_j}$ and $\hat{B}_j = g^{\varrho_j}\mathsf{sn}^{-e_j}$ for $j = [1, v]$
  - $\hat{c} = \mathcal{H}(\mathsf{ct}, \hat{Z}, \hat{R}_1, \hat{R}_2, \hat{R}_3, \hat{A}_1, \cdots, \hat{A}_v, \hat{B}_1, \cdots, \hat{B}_v)$
  - If $c \neq \hat{c}$ or $U^{\varrho_a} \neq \hat{R}_2 \hat{Z}^{\hat{c}}$ then return $\perp$
  - Return $m = \mathsf{ct}_2/\hat{Z}$

$\pi_{\mathsf{upd}} = \mathsf{PoK}\{(r, r', x) :\ V' = g_x g_{v+3}^{r} h^{r'} \wedge g_x \in \{g_1, \cdots, g_v\}\}$

- $\mathsf{Prv}(r, r', x)$:
  - $\theta_x, \theta'_x, \left\{e_j, \theta_j, \theta'_j\right\}_{j=1, j\neq x}^{v} \xleftarrow{\$} \mathbb{Z}_p^*$
    - $A_x = g_{v+3}^{\theta_x} h^{\theta'_x}$
    - For $j \neq x$, $\varrho_j = \theta_j + e_j \cdot r$, $\varrho'_j = \theta'_j + e_j \cdot r'$ and $A_j = g_{v+3}^{\varrho_j} h^{\varrho'_j}(g_j/V')^{e_j}$
  - $c = \mathcal{H}(V', A_1, \cdots, A_v)$, $e_x = c - \sum_{j\neq x} e_j$, $\varrho_x = \theta_x + e_x \cdot r$, $\varrho'_x = \theta'_x + e_x \cdot r'$
  - Return $\pi_{\mathsf{upd}} = (\left\{e_j, \varrho_j, \varrho'_j\right\}_{j=1}^{v})$
- $\mathsf{Ver}(V', \pi_{\mathsf{upd}})$:
  - Parse $\pi_{\mathsf{upd}} = (\left\{e_j, \varrho_j, \varrho'_j\right\}_{j=1}^{v})$
  - $c = \sum_{j=1}^{v} e_j$
  - $\hat{A}_j = g_{v+3}^{\varrho_j} h^{\varrho'_j}(g_j/V')^{e_j}$ for $j \in [1, v]$
  - $\hat{c} = \mathcal{H}(V', \hat{A}_1, \cdots, \hat{A}_v)$
  - If $c = \hat{c}$ then return 1 else return 0

Fig. 2: Specifications of zero-knowledge proofs utilised in pRate.

5.2 Further remarks and extensions

In the following we provide several remarks regarding the functionality and design rationale of our scheme.

Timestamps. In pRate, each score includes a timestamp t which indicates when the score was updated last. Upon advertising the user can choose not to disclose the exact time t but to provide a zero-knowledge proof that their score is recent. We do not enforce each user to have the most recent reputation credential, otherwise it would significantly limit the flexibility on when users would need to submit and update their ratings. However, users who have not been rated for a longer period of time could possibly be disadvantaged because of that. In that case, these users can ask the RM to update the timestamp in their reputation credential without disclosing or changing their scores. For this, the RM chooses a fresh timestamp $\tilde{t}$, picks $\tilde{e}, \tilde{s} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\tilde{C} = (R \cdot g_{v+1}^{\tilde{t}-t} \cdot g_{v+3}^{\tilde{s}})^{\frac{1}{\gamma+\tilde{e}}}$ and sends $(\tilde{e}, \tilde{t}, \tilde{s}, \tilde{C})$ to the corresponding user.

Anonymous advertisements. The one-time advertisements published by users prior to each new transaction are anonymous and cannot be linked to the same publisher. Although this provides strong privacy protection, a malicious user may generate a large amount of advertisements to consume resources of online platforms. The RM can restrict users to publish no more than n advertisements in some period of time (e.g., one day) by publishing a set of generators $\{d_1, \cdots, d_n\}$ that would be valid for that period and requiring users to use these d_i for generation of their advertising identifiers $\mathsf{aid} = (d_i, d_i^k)$ during that period. Note that any user who generates more advertisements than allowed by the RM would become linkable.

Secure channels for token exchange. We require that rating tokens are exchanged over a secure channel that must be setup between the two prospective transaction partners. Note that these partners do not know each other identities since their reputation advertisements are anonymous. For this purpose we can let each user choose a temporary private-public key pair and include the corresponding public key as part of the published advertisement information m. These temporary keys can then be used to execute any standard secure channel establishment protocol to create an authenticated and confidential channel (see [26] for the property of such channels) over which parties would exchange their tokens. The authentication property in this case would imply that the channel is established between the two original yet anonymous advertisers. Communication between two anonymous users can be established through the platform on which the advertisements are published and to which users could connect anonymously via Tor [23] or with the help of distributed ledger techniques as in [32].

Accountability. In the rating accumulation algorithm $\mathsf{RateAcc}$, the RM learns the identities of the rater and ratee (without being able to link them to the advertisements containing transaction details). This can be useful in the detection of conventional attacks against reputation systems [25] such as Sybil attacks, self-promotion attacks where a dishonest user arbitrarily creates rating tokens to increase their own reputation score, and ballot stuffing attacks where dishonest users collude to conduct fake transactions and ratings to improve their reputation scores. The RM can detect such malicious behaviours heuristically, for example, when a user gets an unusual large amount of ratings within a short time period or submits too many ratings. If the RM notices any suspicious activity, the RM can investigate, e.g., request users to provide supporting documents to show that these ratings are based on real transactions, and punish misbehaving users.

Batch accumulation and unlinkability. In the rating accumulation algorithm $\mathsf{RateAcc}$, the RM updates user's j reputation credential and sends a update aux to the corresponding user j. We observe that this part does not have to be performed immediately for each new rating that the RM receives for user j. In our scheme this process can be delayed and performed in a batch in order to reduce the overhead from the reputation update. More precisely, the RM can accumulate multiple ratings in a batch by multiplying all ciphertexts into a single product $V = \prod_\ell V_\ell$ and produce a single update information.

When updating a single rating $V_\ell = g_{x_\ell} g_{v+3}^{r_\ell}$, the ratee uses the update token $\mathsf{UT}_\ell = r_\ell$ to extract g_{x_ℓ} from V_ℓ. The update token UT_ℓ is linked to the serial number $\mathsf{sn}_\ell = g^{r_\ell}$ from the token exchange session. We stress that this link does not compromise rater's anonymity because of the anonymous (one-time) advertisement that was used in the token exchange session. Therefore, the ratee is not able to link any previous or future ratings produced by the same rater. In practice, there might be scenarios where some side channel information could leak the identity of the user, e.g., in Airbnb, a guest would meet the home owner in person. For these applications, we can break the link between the token exchange session and the submitted rating by using a more sophisticated batch accumulation technique that applies additional randomisation as described in the following.

The basic idea is to let the RM randomly blind the ratings and give the ratee a decryption key to remove this randomness from the aggregated ratings. The ratee will no longer be able to learn that a rating value x_ℓ is linked to the serial number sn_ℓ and only learn their aggregated value of $\{x_\ell\}_\ell$. Below we describe how the RM randomises the ratings and how the ratee removes this randomness prior to updating its reputation score. The part for updating $\mathsf{reg}[j], C, R$ is the same as before and is omitted here.

- When a user i submits a rating $\delta_\ell = (\mathsf{sn}_\ell, V_\ell, \mathsf{ct}_\ell, \mathsf{ct}'_\ell, \mathsf{sn}'_\ell, \pi'_{\mathsf{tok},\ell}, \pi_{\mathsf{sub},\ell})$ to the RM, the user additionally picks a randomiser $r'_\ell \xleftarrow{\$} \mathbb{Z}_p^*$, computes $V'_\ell = V_\ell \cdot h^{r'_\ell}$ and a proof $\pi_{\mathsf{upd},\ell} = \mathsf{PoK}\{(r_\ell, r'_\ell, x_\ell) :\ V'_\ell = g_{x_\ell} g_{v+3}^{r_\ell} h^{r'_\ell} \wedge g_{x_\ell} \in \{g_1, \cdots, g_v\}\}$ and sends $(r'_\ell, V'_\ell, \pi_{\mathsf{upd},\ell})$ to the RM. The details of $\pi_{\mathsf{upd},\ell}$ are given in Figure 2.
- After the RM receives n ratings (possibly from different raters) for some user j, it can accumulate these ratings together and update user's j reputation credential once. For this, the RM computes $r' = \sum_\ell r'_\ell$, $H = h^{r'}$, $\pi_{\mathsf{dec}} = \mathsf{PoK}\{(r') : H = h^{r'}\}$, and sends $(\{V'_\ell, \pi_{\mathsf{upd},\ell}\}_\ell, \{\mathsf{sn}_\ell\}_\ell, H, \pi_{\mathsf{dec}})$ to user j. The details of π_{dec} are given in Figure 2.
- User j computes $V' = \prod_\ell V'_\ell$ and eliminates the randomisers $\{r'_\ell\}_\ell$ by computing $V = V'/H (= \prod_\ell V_\ell)$. Further, user j finds a set of update tokens $\{\mathsf{UT}_\ell = r_\ell\}_\ell$ corresponding to the serial numbers $\{\mathsf{sn}_\ell\}_\ell$ and removes the randomisers $\{r_\ell\}_\ell$ by computing $r = \sum_\ell r_\ell$ and $M = V/g_{v+3}^r (= \prod_\ell g_{x_\ell})$. User j can then use a brute-force approach to find $(m_1, \cdots, m_v)$ s.t. $M = g_1^{m_1} \cdots g_v^{m_v}$ and $n = m_1 + \cdots + m_v$, and, finally, update own reputation score as $\mathsf{scr}[j] = (n_1 + m_1, \cdots, n_v + m_v, \tilde{t})$.

We remark that the total number of possible combinations for $(m_1, \cdots, m_v)$ is C_{n+v-1}^{v-1} which is a polynomial of degree v and is feasible to brute-force. For example, if $n = 20$ and $v = 5$, then $C_{n+v-1}^{v-1} = 10626$. Since generators $g_1, \cdots, g_v$ are randomly chosen, the tuple $(m_1, \cdots, m_v)$ that satisfies the above conditions is unique with overwhelming probability. Otherwise, if there is another tuple $(m'_1, \cdots, m'_v)$ for which $M = g_1^{m'_1} \cdots g_v^{m'_v}$ then the equation $g_1^{m_1 - m'_1} \cdots g_v^{m_v - m'_v} = 1$ can be used to find a non-trial relation between $g_1, \cdots, g_v$ and break the DL assumption.

The above technique achieves unlinkability between the token exchange sessions and submitted ratings based on the following argument: Let $T_b = (\mathsf{pp}, x_0, x_1, r_0, r_1, V'_0 = g_{x_b} g_{v+3}^{r_0} h^{r'_0}, V'_1 = g_{x_{1-b}} g_{v+3}^{r_1} h^{r'_1}, H = h^{r'_0 + r'_1})$ with $r'_0, r'_1 \xleftarrow{\$} \mathbb{Z}_p^*$ for $b = 0, 1$. In T_0, V'_0 encrypts the rating value x_0 and V'_1 encrypts the rating value x_1, while in T_1, V'_0 encrypts the rating value x_1 and V'_1 encrypts the rating value x_0. We can easily see that $H = V'_0 V'_1 / (g_{x_0} g_{x_1} g_{v+3}^{r_0} g_{v+3}^{r_1})$ holds for both T_0 and T_1. Since r'_0, r'_1 are chosen uniformly at random, T_0 and T_1 have the same distribution.

5.3 Performance analysis

In the following we evaluate the computational costs of pRate algorithms and sizes of utilized zero-knowledge proofs. We start with the latter.

Size of zero-knowledge proofs. All zero-knowledge proofs used in pRate are short as can be observed based on the summary in Table 1, where v is the rating score and $\ell = \sum_i \ell_i$ is the size of the intervals in predicate $P = \wedge_i (\psi_i \in [0, 2^{\ell_i}))$ (see Appendix B for details), and both can be seen as small constants. Furthermore, the size of the proof π_{rep} in the advertisement can be further reduced to $(v + 11)\mathbb{Z}_p + (2\lceil \log \ell \rceil + 5)\mathbb{G}_1$ using an inner-product proof according to [14].

We illustrate with a concrete example based on a five-star rating scheme, i.e. $v = 5$. Suppose, Alice has a reputation score $\mathsf{scr} = (n_1, n_2, n_3, n_4, n_5, t) = (9, 2, 11, 30, 328, 6940)$ where $t = 6940$ is the number of days elapsed from 1 Jan 2000 to 1 Jan 2019. Alice can prove the following statements about her score:

- The number of 1-star, 2-star and 3-star ratings is less than 16, i.e., $n_1, n_2, n_3 \in [0, 2^4)$. The average score is higher than 4.6, i.e., $(n_1 + 2n_2 + 3n_3 + 4n_4 + 5n_5)/(n_1 + n_2 + n_3 + n_4 + n_5) > 4.6$ which can be proved by showing $n_4, n_5 \in [0, 2^{10})$ and $(-18n_1 - 13n_2 - 8n_3 - 3n_4 + 2n_5) \in [0, 2^{10})$.
- The score was updated no earlier than 1 Oct 2018, i.e., $t > 6848$ where 6848 is the number of days elapsed from 1 Jan 2000 to 1 Oct 2018. This can be proved using $t \in [0, 2^{13})$ and $t - 6848 \in [0, 2^{13})$.

This leads to $\ell = 4 * 3 + 10 * 3 + 13 * 2 = 68$ and $\lceil \log \ell \rceil = 7$.

Zero-knowledge proofs	Numbers of group elements
π_{id}	$3\mathbb{Z}_p$
π_{enc}	$3\mathbb{Z}_p$
π_{dec}	$2\mathbb{Z}_p$
π_{tok}	$3\mathbb{Z}_p + 2\mathbb{G}_1$
π_{sub}	$(2v+2)\mathbb{Z}_p + 2\mathbb{G}_1$
π_{upd}	$(3v)\mathbb{Z}_p$
π_{rep}	$(v+2\ell+9)\mathbb{Z}_p + 3\mathbb{G}_1$

Table 1: Sizes of zero-knowledge proofs in pRate.

Operations	Numbers of group operations
Join	$(v+4)mul_{\mathbb{G}_1} + (v+8)exp_{\mathbb{G}_1} + 2pairing$
Issue	$(v+6)mul_{\mathbb{G}_1} + (v+7)exp_{\mathbb{G}_1} + 2pairing$
RepAds	$(4\ell+1)mul_{\mathbb{G}_1} + (4\ell+3)exp_{\mathbb{G}_1} + (v+4)mul_{\mathbb{G}_T} + (v+5)exp_{\mathbb{G}_T}$
RepVer	$(v+2\ell+6)mul_{\mathbb{G}_1} + (v+2\ell+10)exp_{\mathbb{G}_1} + 1mul_{\mathbb{G}_T} + 2pairing$
Token	$12mul_{\mathbb{G}_1} + 28exp_{\mathbb{G}_1}$
RateGen	$9mul_{\mathbb{G}_1} + (2v+15)exp_{\mathbb{G}_1}$
RateAcc	$18mul_{\mathbb{G}_1} + (4v+21)exp_{\mathbb{G}_1}$
Upd	$20mul_{\mathbb{G}_1} + (4v+23)exp_{\mathbb{G}_1} + 2pairing$

Table 2: Computational costs of pRate algorithms.

Computational costs. We summarize the amount of computations for each pRate algorithm in Table 2, where $mul_{\mathbb{G}_1}$ and $mul_{\mathbb{G}_T}$ denote scalar multiplications in $\mathbb{G}_1$ and $\mathbb{G}_T$, respectively; $exp_{\mathbb{G}_1}$ and $exp_{\mathbb{G}_T}$ are exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, respectively; RepAds and RepVer denote time-consuming pairing operations which can be optimized further (as discussed in Appendix B.1).

6 Security analysis of pRate

pRate is designed to minimize the trust put on the RM. In particular, to ensure that (1) the RM cannot learn any rating values submitted by the users, (2) the RM cannot link submitted ratings to the published anonymous advertisements, and (3) all data sent by the RM to the intended recipients is verifiable. The security of our pRate scheme is established formally in Theorems 1 to 5 based on the properties of anonymity, rating secrecy, and the three flavours of unforgeability from Section 3. The formal proofs of all theorems are provided in Appendix C. In the following we provide only high-level intuition for the security of pRate. We note that all security properties hold in the random oracle model due to the use of non-interactive zero-knowledge proofs based on the well-known Fiat-Shamir transformation.

Theorem 1. *The pRate scheme is anonymous under the XDH assumption.*

An advertisement $(\mathsf{aid}, \pi_{\mathsf{rep}})$ created by a user is fully anonymous due to the use of randomly chosen one-time identifiers $\mathsf{aid} = (d, d^k)$ and the zero-knowledge property of π_{rep}. The token exchange protocol is performed over a secure channel so that the RM cannot link the exchanged tokens to the ratings that it receives. The ciphertexts ct and ct' encrypting the identities of participating users can only be decrypted by the RM so that users remain anonymous to each other. The anonymity holds even when the RM's issuing key ik becomes compromised. The proofs π_{tok} used in rating tokens and π_{sub} used in ratings are constructed based on techniques from CPS-EG which allow us to create a decryption oracle without knowing the RM's opening key ok.

Theorem 2. *The pRate scheme is rating-secret under the XDH assumption.*

Each rating is encrypted in $V = g_x g_{v+3}^r$ using random r that is only known to the rater and the ratee. The zero-knowledge proof π_{sub} ensures that V is correctly formed without leaking any information about x. The RM accumulates ciphertexts V to the ratee's reputation credential without learning the value of x. As long as the rater and the ratee are honest, their rating stays confidential. This holds even if the RM's issuing and opening keys become compromised.

Theorem 3. *The pRate scheme is advertisement-unforgeable under the q-SDH assumption.*

Note that only users in possession of valid reputation credentials issued to them by the RM can generate verifiable advertisements. The unforgeability of advertisements relies on the unforgeability of the BBS+ signature scheme and holds for honest users, even if the RM's opening key becomes compromised.

Theorem 4. *The pRate scheme is ratee-unforgeable under the DL assumption.*

Unforgeability of rating tokens RT, computed using the long-term secret key k of the ratee, relies on the zero-knowledge and soundness properties of the proofs π_{tok} used in the token exchange protocol and π_{id} used in the registration protocol. Note that in case of successful forgery, the forking lemma can be used to extract k. The ratee-unforgeability property holds for honest ratees, in presence of the possibly corrupted RM.

Theorem 5. *The pRate scheme is rater-unforgeable under the DL assumption.*

Unforgeability of ratings δ, computed using the long-term secret key k of the rater, relies on the zero-knowledge and soundness properties of the proofs π_{sub} used in the rating generation algorithm and π_{id} used in the registration protocol. Note that in case of successful forgery, the forking lemma can be used to extract k. The rater-unforgeability property holds for honest raters, in presence of the possibly corrupted RM.

7 Other related work

In terms of privacy, pRate is superior to a number of existing reputation schemes where anonymity of users is provided without considering the secrecy of their reputation scores. The scheme in [22] adopts controlled anonymity and cluster filtering to leverage against the effects of unfair ratings and discriminating seller. The system relies on a trusted third party called marketplace to publish the estimated reputation of buyers and sellers and assigns them pseudonyms to perform transactions. PERM [6] provides reputation-based blacklisting which enables a service provider to score users' anonymous sessions and deny access to users with insufficient reputation. A user's reputation score is uniquely identified with a serial number which will be revealed after the service provider updates the score and generates a new serial number. Therefore the rating by the service provider must be performed sequentially, i.e., one session after another. The work in [21] studies relations on several privacy definitions for reputation systems and presents a reputation function that can satisfy k-anonymity and rating secrecy. An anonymous reputation system based on pseudonymous system and e-cash is described in [5]. An authority called Bank keeps the record of each user's reputation score. Two users communicate with each other under their one-time pseudonyms where one user can rate the other user by transferring a certain amount of repcoins assigned by the Bank via e-cash. This system lacks accountability since users can create an arbitrary number of pseudonyms which are not registered with any authority. A reputation framework for participatory sensing applications is proposed in [20], where a user reports sensor readings to an application server and the server computes a score by evaluating the accuracy of the readings. Each user uses a pseudonym for reporting readings within a certain period and transfers the gained score to the next pseudonym when the next time period starts while preventing attackers from linking these pseudonyms. Reputation systems proposed in [9–11] allow each user to anonymously rate a product at most once. If a user rates the same product multiple times, his anonymity will be broken because these ratings are linkable. The system in [11] is based on group signatures with linkability while the scheme in [9] combines anonymous credentials with a reputation system. These systems focus on protecting anonymity for the rater but not for the ratee and they do not consider how to manage and protect reputation scores. Similar considerations apply to the scheme in [10] which mainly focuses on the universal composability of reputation systems. An anonymous reputation system which gives users rewards for submitting useful comments is presented in [15]. Users can publish their assessment opinions which can then be endorsed by other users such that the original rater receives some reward upon receiving a threshold number of endorsements.

8 Conclusion

In this paper we introduced pRate, a novel privacy-preserving reputation system, where scores are computed based on the (aggregated) number(s) of stars that users receive from their raters. The scheme is managed by a possibly untrusted reputation manager who can register users and assist ratees in updating their reputation scores, yet without learning these scores. In addition to ensuring the secrecy of the ratings, a distinctive feature of pRate over prior proposals, is that it hides the identities of raters and ratees from each other during the transaction and rating stages. pRate can be extended with a randomised batch accumulation technique that will further prevent the ratee from linking the received ratings to the corresponding transactions, thus offering even stronger privacy protection for the ratee. We note that pRate is widely independent of the actual transactions that occur between users and eventual payments associated with these transactions. As such pRate can be used in combination with many other approaches for anonymous transaction processing and payment.

References

1. How Online Reviews Will Impact Your Practice in 2018. `https://virayo.com/online-reputation-management/importance-of-online-reviews/`.
2. Yelp accused of bullying businesses into paying for better reviews. `http://www.cbc.ca/news/business/yelp-accused-of-bullying-businesses-into-paying-for-better-reviews-1.2899308`.
3. Yelp Accused of Extortion. `https://www.wired.com/2010/02/yelp-sued-for-alleged-extortion/`.
4. Yelp Accused Of Hiding Positive Reviews For Non-Advertiser. `https://dfw.cbslocal.com/2018/01/09/yelp-accused-hiding-positive-reviews-non-advertiser/`.
5. E. Androulaki, S. G. Choi, S. M. Bellovin, and T. Malkin. Reputation systems for anonymous networks. In *Privacy Enhancing Technologies*, pages 202–218, 2008.
6. M. H. Au and A. Kapadia. PERM: Practical Reputation-based Blacklisting Without TTPS. CCS '12, pages 929–940, 2012.
7. M. H. Au, W. Susilo, and Y. Mu. Constant-Size Dynamic k-TAA. pages 111–125, 2006.
8. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT*, pages 614–629, 2003.
9. K. Bemmann, J. Blömer, J. Bobolz, H. Bröcher, D. Diemert, F. Eidens, L. Eilers, J. Haltermann, J. Juhnke, B. Otour, L. Porzenheim, S. Pukrop, E. Schilling, M. Schlichtig, and M. Stienemeier. Fully-featured anonymous credentials with reputation system. ARES, 2018.
10. J. Blömer, F. Eidens, and J. Juhnke. Practical, anonymous, and publicly linkable universally-composable reputation systems. CT-RSA, 2018.
11. J. Blömer, J. Juhnke, and C. Kolb. Anonymous and publicly linkable reputation systems. In *Financial Cryptography and Data Security*, pages 478–488, 2015.
12. D. Boneh, X. Boyen, and H. Shacham. Short Group Signatures. In *CRYPTO*, 2004.
13. J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth. Foundations of fully dynamic group signatures. In *ACNS*, pages 117–136, 2016.
14. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE S&P*, pages 319–338, 2018.
15. N. Busom, R. Petrlic, F. Sebé, C. Sorge, and M. Valls. A privacy-preserving reputation system with user rewards. *Journal of Network and Computer Applications*, 80:58–66, 2017.
16. J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-Cash. EUROCRYPT'05, pages 302–321, 2005.
17. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, pages 56–72, 2004.
18. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups (extended abstract). CRYPTO '97, pages 410–424, 1997.
19. A. S. Y. Cheung and W. Schulz. Reputation protection on online rating sites. *Stanford Technology Law Review*, 2018.
20. D. Christin, C. R. kopf, M. Hollick, L. A. Martucci, and S. S. Kanhere. IncogniSense: An anonymity-preserving reputation framework for participatory sensing applications. *Pervasive and Mobile Computing*, 9(3):353 – 371, 2013.
21. S. Clauß, S. Schiffner, and F. Kerschbaum. K-anonymous reputation. ASIA CCS, 2013.
22. C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. EC'00, pages 150–157, 2000.
23. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *In 13'th USENIX Security*, 2004.
24. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, 2008.
25. O. Hasan. A Survey of Privacy Preserving Reputation Systems. Technical Report. LIRIS UMR 5205CNRS, 2017.
26. T. Jager, F. Kohlar, S. Schäge, and J. Schwenk. On the security of TLS-DHE in the standard model. In *CRYPTO 2012*, volume 7417, pages 273–293, 2012.
27. F. Kerschbaum. A verifiable, centralized, coercion-free reputation system. WPES '09, pages 61–70, 2009.
28. T. Minkus and K. W. Ross. I know what you're buying: Privacy breaches on ebay. In E. De Cristofaro and S. J. Murdoch, editors, *PETS*, pages 164–183, 2014.
29. A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (S&P 2008)*, pages 111–125, 2008.
30. R. Petrlic, S. Lutters, and C. Sorge. Privacy-preserving reputation management. SAC '14, pages 1712–1718, 2014.
31. P. Resnick and R. Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. Volume 11 of Advances in Applied Microeconomics.
32. E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.
33. Y. Seurin and J. Treger. A robust and plaintext-aware variant of signed elgamal encryption. In *CT-RSA*, pages 68–83, 2013.
34. T. Teubner, F. Hawlitschek, and D. Dann. Price determinants on Airbnb: How reputation pays off in the sharing economy. *Journal of Self-Governance and Management Economics*, 5(4):53–80, 2017.
35. G. Zervas, D. Proserpio, and J. Byers. A first look at online reputation on airbnb, where every stay is above average. In *SSRN Working Paper 2554500*, 2015.
36. E. Zhai, D. I. Wolinsky, R. Chen, E. Syta, C. Teng, and B. Ford. Anonrep: Towards tracking-resistant anonymous reputation. In *NSDI 16*, pages 583–596, 2016.

A Definitions of correctness and security

The correctness and security properties are formulated via experiments where an adversary is given access to certain oracles defined in Figure 3.

Correctness The correctness guarantees that 1) an advertisement produced by honest users are accepted by the verification algorithm RepVer; 2) ratings constructed by honest users are accepted by the accumulation algorithm RateAcc; 3) updates from honest RM are accepted by the update algorithm Upd. Formally, a pRate scheme is *correct* if $\mathsf{Adv}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda) := \mathsf{Pr}[\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda) = 1]$ is negligible, where $\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda)$ is defined below:

$\underline{\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A}}(1^\lambda)}$:

- $(\mathsf{ik}, \mathsf{ok}, \mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
- $\mathsf{HU} = \mathsf{BU} = \mathsf{ChU} = \mathsf{AL} = \mathsf{RL} = \mathsf{UL} = \emptyset$
- $(i_0, i_1, m, P, m', P', x) \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU},\mathsf{RReg},\mathsf{RepAds},\mathsf{Token},\mathsf{SndTok},\mathsf{RateGen},\mathsf{RateUpd}}(1^\lambda, \mathsf{pp})$
- If $i_0 \notin \mathsf{HU}$ or $i_1 \notin \mathsf{HU}$ or $P(\mathsf{scr}[i_0]) = 0$ or $P'(\mathsf{scr}[i_1]) = 0$ or $x \notin [1, v]$ then return 0
- $(\mathsf{aid}, \pi_{\mathsf{rep}}) \xleftarrow{\$} \mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[i_0], \mathsf{scr}[i_0], m, P)$
- If $\mathsf{RepVer}(\mathsf{pp}, \mathsf{aid}, \pi_{\mathsf{rep}}, m, P) = 0$ then return 1
- $(\mathsf{aid}', \pi'_{\mathsf{rep}}) \xleftarrow{\$} \mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[i_1], \mathsf{scr}[i_1], m', P')$
- $((\mathsf{sn}, \mathsf{sn}', \mathsf{RT}, \mathsf{UT}), (\mathsf{sn}', \mathsf{sn}, \mathsf{RT}', \mathsf{UT}')) \xleftarrow{\$} (\mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_0], \mathsf{aid}, \mathsf{aid}'), \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_1], \mathsf{aid}', \mathsf{aid}))$
- $\delta \xleftarrow{\$} \mathsf{RateGen}(\mathsf{pp}, \mathsf{urep}[i_0], \mathsf{RT}, x)$
- $\theta \xleftarrow{\$} \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$
- If $\theta = \perp$ then return 1
- Parse $\theta = (i'_0, i'_1, \mathsf{aux})$
- If $i'_0 \neq i_0$ or $i'_1 \neq i_1$ then return 1
- If $\mathsf{Upd}(\mathsf{pp}, \mathsf{urep}[i_1], \mathsf{scr}[i_1], \mathsf{UT}', \mathsf{aux}) = 0$ then return 1
- Return 0

Anonymity Following [8], the anonymity is defined in a way that an adversary does not need to recover a user's identity but only distinguish which of two users of its choice produced an advertisment, engaged in a token generation session and generated a rating. A pRate scheme is *anonymous* if $\mathsf{Adv}^{\mathsf{anony}}_{\mathcal{A}}(1^\lambda) := |\mathsf{Pr}[\mathsf{Exp}^{\mathsf{anony}}_{\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2}|$ is negligible, where $\mathsf{Exp}^{\mathsf{anony}}_{\mathcal{A}}(1^\lambda)$ is an experiment defined below.

$\underline{\mathsf{Exp}^{\mathsf{anony}}_{\mathcal{A}}(1^\lambda)}$:

- $(\mathsf{ik}, \mathsf{ok}, \mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
- $\mathsf{HU} = \mathsf{BU} = \mathsf{ChU} = \mathsf{AL} = \mathsf{RL} = \mathsf{UL} = \emptyset$
- $(i_0, i_1, j, m, P, x, \mathsf{St}) \xleftarrow{\$} \mathcal{A}_0^{\mathsf{AddU},\mathsf{WReg},\mathsf{CrptU},\mathsf{RepAds},\mathsf{Token},\mathsf{SndTok},\mathsf{RateGen},\mathsf{RateUpd}}(1^\lambda, \mathsf{pp}, \mathsf{ik})$
- If $i_0 \notin \mathsf{HU}\backslash\mathsf{BU}$ or $i_1 \notin \mathsf{HU}\backslash\mathsf{BU}$ or $P(\mathsf{scr}[i_0]) = 0$ or $P(\mathsf{scr}[i_1]) = 0$ or $x \notin [1, v]$ then return 0
- $\mathsf{ChU} = \mathsf{ChU} \cup \{i_0, i_1\}$
- $b \xleftarrow{\$} \{0, 1\}$
- $(\mathsf{aid}^*, \pi^*_{\mathsf{rep}}) \xleftarrow{\$} \mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[i_b], \mathsf{scr}[i_b], m, P)$
- $(\mathsf{aid}, \pi_{\mathsf{rep}}) \xleftarrow{\$} \mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{scr}[j], m, P)$
- $((\mathsf{sn}, \mathsf{sn}', \mathsf{RT}, \mathsf{UT}), (\mathsf{sn}', \mathsf{sn}, \mathsf{RT}', \mathsf{UT}')) \xleftarrow{\$} \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_b], \mathsf{aid}^*, \mathsf{aid}), \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{aid}, \mathsf{aid}^*))$
- $\mathsf{UL}[\mathsf{lu}{+}{+}] = (\mathsf{sn}', \mathsf{UT})$; $\mathsf{UL}[\mathsf{lu}{+}{+}] = (\mathsf{sn}, \mathsf{UT}')$
- $\delta^* \xleftarrow{\$} \mathsf{RateGen}(\mathsf{pp}, \mathsf{urep}[i_b], \mathsf{RT}, x)$
- $\sigma^* = (\mathsf{aid}^*, \pi^*_{\mathsf{rep}}, \mathsf{sn}', \mathsf{RT}', \mathsf{UT}', \delta^*)$
- $\hat{b} \xleftarrow{\$} \mathcal{A}_1^{\mathsf{AddU},\mathsf{WReg},\mathsf{CrptU},\mathsf{RepAds},\mathsf{Token},\mathsf{SndTok},\mathsf{RateGen},\mathsf{RateUpd}}(\mathsf{St}, \sigma^*)$
- Return $b = \hat{b}$

Rating-secrecy Rating-secrecy is defined in a way that an adversary needs to distinguish which of two rating values of its choice is encrypted in the rating. Formally, a pRate scheme is *rating-secret* if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{secrecy}}(1^\lambda) := |\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{secrecy}}(1^\lambda) = 1] - \frac{1}{2}|$ is negligible, where $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{secrecy}}(1^\lambda)$is an experiment defined below.

```
$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{secrecy}}(1^\lambda):}$
– $(\mathsf{ik}, \mathsf{ok}, \mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
– $\mathsf{HU} = \mathsf{BU} = \mathsf{ChU} = \mathsf{AL} = \mathsf{RL} = \mathsf{UL} = \emptyset$
– $(x_0, x_1, k, \mathsf{St}) \xleftarrow{\$} \mathcal{A}_0^{\mathsf{AddU},\mathsf{RReg},\mathsf{CrptU},\mathsf{RepAds},\mathsf{Token},\mathsf{RateGen},\mathsf{RateAcc},\mathsf{RateUpd}}(1^\lambda, \mathsf{pp}, \mathsf{ik}, \mathsf{ok})$
– If $x_0 \notin [1, v]$ or $x_1 \notin [1, v]$ or $\mathsf{RL}[k] = \perp$
– Parse $\mathsf{RL}[k] = (i, \mathsf{sn}, \mathsf{RT}, \mathit{flag})$
– If $i \notin \mathsf{HU}$ or $\mathit{flag} = 1$ then return $\perp$
– $b \xleftarrow{\$} \{0, 1\}$
– $\delta^* \xleftarrow{\$} \mathsf{RateGen}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{RT}, x_b)$
– $\mathsf{RL}[k] = (i, \mathsf{sn}, \mathsf{RT}, 1)$
– $\hat{b} \xleftarrow{\$} \mathcal{A}_1^{\mathsf{AddU},\mathsf{RReg},\mathsf{Token},\mathsf{RateGen},\mathsf{RateAcc},\mathsf{RateUpd}}(\mathsf{St}, \delta^*)$
– Return $b = \hat{b}$
```

Advertisement-unforgeability This ensures that an adversary cannot produce an advertisement that cannot be traced back to an active member. The adversary generates an advertisement and wins the experiment if the advertisement cannot be used to generate a rating token, submit a rating and accumulate a rating for any valid user. Formally, a pRate scheme is *advertisement-unforgeable* if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{trace}}(1^\lambda) := |\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ad-unforge}}(1^\lambda) = 1]|$ is negligible, where $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ad-unforge}}(1^\lambda)$ is an experiment defined below.

```
$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ad-unforge}}(1^\lambda):}$
– $(\mathsf{ik}, \mathsf{ok}, \mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
– $\mathsf{HU} = \mathsf{BU} = \mathsf{ChU} = \mathsf{AL} = \mathsf{RL} = \mathsf{UL} = \emptyset$
– $(\mathsf{aid}, \pi_{\mathsf{rep}}, m, P, n, x) \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU},\mathsf{RReg},\mathsf{CrptU},\mathsf{RepAds},\mathsf{Token},\mathsf{SndTok},\mathsf{RateGen},\mathsf{RateUpd}}(1^\lambda, \mathsf{pp}, \mathsf{ok})$
– If $\mathsf{RepVer}(\mathsf{pp}, \mathsf{aid}, \pi_{\mathsf{rep}}, m, P) = 0$ or $\mathsf{AL}[n] = \perp$ or $x \notin [1, v]$ then return 0
– Parse $\mathsf{AL}[n] = (j, \mathsf{aid}')$
– If $\nexists i \in \mathsf{HU}$ s.t. $\theta = (*, i, *)$ where $\theta$ is computed from
    • $((\mathsf{sn}, \mathsf{sn}', \mathsf{RT}, \mathsf{UT}), (\mathsf{sn}', \mathsf{sn}, \mathsf{RT}', \mathsf{UT}')) \xleftarrow{\$} (\mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{aid}, \mathsf{aid}'), \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{aid}', \mathsf{aid}))$
    • $\delta \xleftarrow{\$} \mathsf{RateGen}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{RT}', x)$
    • $\theta \xleftarrow{\$} \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$
  then return 1 else return 0
```

Ratee-unforgeability This ensures that an adversary cannot forge a valid rating token that involves an honest user as ratee unless this user does produce it. A pRate scheme is *ratee-unforgeable* if $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{unforge-ratee}}(1^\lambda) := |\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ratee-unforge}}(1^\lambda) = 1]|$ is negligible, where $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ratee-unforge}}(1^\lambda)$ is an experiment defined below.

```
$\underline{\mathsf{Exp}_{\mathcal{A}}^{\mathsf{ratee-unforge}}(1^\lambda):}$
– $(\mathsf{ik}, \mathsf{ok}, \mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
– $\mathsf{HU} = \mathsf{BU} = \mathsf{ChU} = \mathsf{AL} = \mathsf{RL} = \mathsf{UL} = \emptyset$
– $\delta \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU},\mathsf{RReg},\mathsf{CrptU},\mathsf{RepAds},\mathsf{Token},\mathsf{SndTok},\mathsf{RateGen},\mathsf{RateAcc},\mathsf{RateUpd}}(1^\lambda, \mathsf{pp}, \mathsf{ik}, \mathsf{ok})$
– $\theta \xleftarrow{\$} \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$
– If $\theta = \perp$ then return 0
– Parse $\theta = (i, j, \mathsf{aux})$
– If the following are all true then return 1 else return 0
    • $j \in \mathsf{HU} \backslash \mathsf{BU}$
    • $\mathcal{A}$ does not query $\mathsf{SndTok}(n, *)$ with $\mathsf{AL}[n] = (j, *)$, or $\mathsf{Token}(j_0, j_1)$ with $\mathsf{AL}[j_0] = (j, *)$ or $\mathsf{AL}[j_1] = (j, *)$
```

```
$\mathsf{AddU}(i)$
– If $i \in \mathsf{HU}$ then return $\perp$
– $\mathsf{HU} = \mathsf{HU} \cup \{i\}$; $\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{cont}$
– $\mathsf{St}^i_{\mathsf{Join}} = (\mathsf{pp}, i)$; $\mathsf{St}^i_{\mathsf{Issue}} = (\mathsf{pp}, \mathsf{ik}, i)$
– $(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^i_{\mathsf{Join}}) \xleftarrow{\$} \mathsf{Join}(\mathsf{St}^i_{\mathsf{Join}})$
– While ($\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{cont}$ and $\mathsf{dec}^i_{\mathsf{Join}} = \mathsf{cont}$) Do
    • $(\mathsf{St}^i_{\mathsf{Issue}}, M_{\mathsf{Join}}, \mathsf{dec}^i_{\mathsf{Issue}}) \xleftarrow{\$} \mathsf{Issue}(\mathsf{St}^i_{\mathsf{Issue}}, M_{\mathsf{Issue}})$
    • $(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^i_{\mathsf{Join}}) \xleftarrow{\$} \mathsf{Join}(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{Join}})$
– If $\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{accept}$ then $\mathsf{reg}[i] = \mathsf{St}^i_{\mathsf{Issue}}$
– If $\mathsf{dec}^i_{\mathsf{Join}} = \mathsf{accept}$ then $(\mathsf{urep}[i], \mathsf{scr}[i]) = \mathsf{St}^i_{\mathsf{Join}}$
```

```
$\mathsf{RepAds}(i, m, P)$
– Return $\perp$ if $i \notin \mathsf{HU} \backslash \mathsf{ChU}$
– $(\mathsf{aid}, \pi_{\mathsf{rep}}) \xleftarrow{\$} \mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{scr}[i], m, P)$
– $\mathsf{AL}[\mathsf{la}{+}{+}] = (i, \mathsf{aid})$
– Return $\sigma$
```

```
$\mathsf{CrptU}(i)$
– Return $\perp$ if $i \notin \mathsf{HU} \backslash (\mathsf{BU} \cup \mathsf{ChU})$
– $\mathsf{BU} \xleftarrow{\$} \mathsf{BU} \cup \{i\}$
– Return $(\mathsf{urep}[i], \mathsf{scr}[i])$
```

```
$\mathsf{RateGen}(k, x)$
– Return $\perp$ if $x \notin [1, v]$
– Parse $\mathsf{RL}[k] = (i, \mathsf{sn}, \mathsf{RT}, \mathit{flag})$
– If $\mathit{flag} = 1$ then return $\perp$
– $\delta \xleftarrow{\$} \mathsf{RateGen}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{RT}, x)$
– $\mathsf{RL}[k] = (i, \mathsf{sn}, \mathsf{RT}, 1)$
– Return $(\mathsf{sn}, \delta)$
```

```
$\mathsf{RReg}(i)$:
– Return $\mathsf{reg}[i]$
```

```
$\mathsf{WReg}(i, val)$:
– $\mathsf{WReg}[i] \xleftarrow{\$} val$
```

```
$\mathsf{RateUpd}(\mathsf{sn}, \delta, M_{\mathsf{in}})$
– $\theta \xleftarrow{\$} \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$
– If $\theta = \perp$ return $\perp$
– Parse $\theta = (i, j, \mathsf{aux})$
– Find $k$ s.t. $\mathsf{UL}[k] = (\mathsf{sn}, \mathsf{UT}')$
– If such $k$ exists then return $\mathsf{Upd}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{scr}[j],$
  $\mathsf{UT}', \mathsf{aux})$
– Return $\mathsf{Upd}(\mathsf{pp}, \mathsf{urep}[j], \mathsf{scr}[j], M_{\mathsf{in}}, \mathsf{aux})$
```

```
$\mathsf{SndTok}(j, M_{\mathsf{in}})$
– Parse $M_{\mathsf{in}} = (\mathsf{aid}', \pi'_{\mathsf{rep}}, m, P, M)$
– If $\mathsf{RepVer}(\mathsf{pp}, \mathsf{aid}', \pi'_{\mathsf{rep}}, m, P) = 0$ or $\mathsf{AL}[j] = \perp$ then return $\perp$
– Parse $\mathsf{AL}[j] = (i, \mathsf{aid})$
– If $\mathsf{St}^j_{\mathsf{Token}}$ is undefined then $\mathsf{St}^j_{\mathsf{Token}} = (\mathsf{pp}, \mathsf{urep}[i],$
  $\mathsf{aid}, \mathsf{aid}')$
– $(\mathsf{St}^j_{\mathsf{Token}}, M_{\mathsf{out}}, \mathsf{dec}^j_{\mathsf{Token}}) \xleftarrow{\$} \mathsf{Token}(\mathsf{St}^j_{\mathsf{Token}}, M)$
– If $\mathsf{dec}^j_{\mathsf{Token}} = \mathsf{accept}$ then
    • Parse $\mathsf{St}^j_{\mathsf{Token}} = (\mathsf{sn}, \mathsf{sn}', \mathsf{RT}, \mathsf{UT})$
    • $\mathsf{RL}[\mathsf{lt}{+}{+}] = (i, \mathsf{sn}, \mathsf{RT}, 0)$
    • $\mathsf{UL}[\mathsf{lu}{+}{+}] = (\mathsf{sn}', \mathsf{UT})$
– Return $(M_{\mathsf{out}}, \mathsf{dec}^j_{\mathsf{Token}})$
```

```
$\mathsf{Token}(j_0, j_1)$
– Return $\perp$ if $\mathsf{AL}[j_0] = \perp$ or $\mathsf{AL}[j_1] = \perp$
– Let $\mathsf{AL}[j_0] = (i_0, \mathsf{aid}_0)$ and $\mathsf{AL}[j_1] = (i_1, \mathsf{aid}_1)$
– $((\mathsf{sn}, \mathsf{sn}', \mathsf{RT}, \mathsf{UT}), (\mathsf{sn}', \mathsf{sn}, \mathsf{RT}', \mathsf{UT}')) \xleftarrow{\$} (\mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_0], \mathsf{aid}_0, \mathsf{aid}_1), \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_1], \mathsf{aid}_1, \mathsf{aid}_0))$
– $\mathsf{RL}[\mathsf{lr}{+}{+}] = (i_0, \mathsf{sn}, \mathsf{RT}, 0)$; $\mathsf{UL}[\mathsf{lu}{+}{+}] = (\mathsf{sn}', \mathsf{UT})$
– $\mathsf{RL}[\mathsf{lr}{+}{+}] = (i_1, \mathsf{sn}', \mathsf{RT}', 0)$; $\mathsf{UL}[\mathsf{lu}{+}{+}] = (\mathsf{sn}, \mathsf{UT}')$
```

Fig. 3: Oracles used in security games. The oracles manipulate the following global variables: a set HU of honest users, a set BU of users whose reputation credentials and scores have been revealed to the adversary, a set ChU of users chosen by the adversary in defining anonymity, a list AL of user id and advertising identifier , a list TL of rating and updating-tokens. The sets HU, BU are initially empty and all the entries of the lists $\mathsf{AL}, \mathsf{RL}, \mathsf{UL}$ are initialised to be $\perp$. The length of AL (or RL, UL) is recorded using a global variable la (or lr, lu) which is initially $\mathsf{la} = 0$ ($\mathsf{lr} = \mathsf{lu} = 0$).

Rater-unforgeability This ensures that an adversary cannot forge a valid rating that involves an honest user as rater unless this user does produce it. A pRate scheme is *rater-unforgeable* if $\mathsf{Adv}^{\mathsf{unforge-rater}}_{\mathcal{A}}(1^\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{rater-unforge}}_{\mathcal{A}}(1^\lambda) = 1]|$ is negligible, where $\mathsf{Exp}^{\mathsf{rater-unforge}}_{\mathcal{A}}(1^\lambda)$ is an experiment defined below.

```
$\underline{\mathsf{Exp}^{\mathsf{rater-unforge}}_{\mathcal{A}}(1^\lambda)}$:
– $(\mathsf{ik}, \mathsf{ok}, \mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
– $\mathsf{HU} = \mathsf{BU} = \mathsf{ChU} = \mathsf{AL} = \mathsf{RL} = \mathsf{UL} = \emptyset$
– $\delta \xleftarrow{\$} \mathcal{A}^{\mathsf{AddU}, \mathsf{RReg}, \mathsf{CrptU}, \mathsf{RepAds}, \mathsf{Token}, \mathsf{SndTok}, \mathsf{RateGen}, \mathsf{RateAcc}, \mathsf{RateUpd}}(1^\lambda, \mathsf{pp}, \mathsf{ik}, \mathsf{ok})$
– $\theta \xleftarrow{\$} \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$
– If $\theta = \perp$ then return 0
– Parse $\theta = (i, j, \mathsf{aux})$
– If the following are all true then return 1 else return 0
    • $i \in \mathsf{HU} \backslash \mathsf{BU}$
    • $\mathcal{A}$ does not query $\mathsf{RateGen}(k, \cdot)$ with $\mathsf{RL}[k] = (i, *, *, *)$
```

B Reputation advertisements using Bulletproofs

We can integrate Bulletproofs [14] into reputation advertisements of pRate to prove a predicate P consisting of multiple interval statements of the form $\psi \in [0, 2^\ell - 1]$. This is sufficient for expressing most of the common measurements used in a star rating system, for example, the average score is above a certain value $\mathsf{avg} \geq \hat{v}$ and the score is updated no earlier than a certain time point $t \geq \hat{t}$. An advantage of using Bulletproofs [14] is that it enables us to aggregate multiple interval proofs into one proof with only logarithmic extra overhead. Below we describe how to prove a general form of predicate $P = \bigwedge_{i=1}^{\zeta} \left(\psi_i \in [0, 2^{\ell_i})\right)$ where $\psi_i = \sum_{j=0}^{v} \beta_{ij} n_j$ for $i \in [1, \zeta - 1]$ and β_{ij} are small integer coefficients and $n_0 = 1$ and $\psi_\zeta = t - \tilde{t}$. Let $\ell = \sum_{i=1}^{\zeta} \ell_i$.

B.1 Advertisement generation and verification ($\mathsf{RepAds}, \mathsf{RepVer}$)

The RM adds auxiliary public parameters $h \xleftarrow{\$} \mathbb{G}_1, \mathbf{g}, \mathbf{h} \xleftarrow{\$} \mathbb{G}_1^{\ell}$ which are independent generators for range proof. We use the following notations from Bulletproofs [14]:

– Bold font denotes vectors, e.g., $\mathbf{a} \in \mathbb{F}^{\ell}$ is a vector with ℓ elements $a_1, \cdots, a_\ell \in \mathbb{F}$;
– $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{\ell} a_i \cdot b_i$ is the inner product between two vetors and $\mathbf{a} \circ \mathbf{b} = (a_1 \cdot b_1, \cdots, a_n \cdot b_n)$ is the Hadamard product (entry wise multiplication of two vectors);
– Given $\mathbf{g} = (g_1, \cdots, g_\ell) \in \mathbb{F}^{\ell}$ and $\mathbf{a} = (a_1, \cdots, a_\ell) \in \mathbb{Z}_p^*$, we write $\mathbf{g}^{\mathbf{a}} = \prod_{i=1}^{\ell} g_i^{a_i}$;
– For $y \in \mathbb{Z}_p^*$, $\mathbf{y}^{\ell} = (1, y, y^2, \cdots, y^{\ell-1})$ is a vector containing the first ℓ powers of y;
– For a vector $\mathbf{a} \in \mathbb{F}^{\ell}$, $\mathbf{a}_{[i,j]} = (a_{i+1}, \cdots, a_j)$ denotes slices of the vector.

Advertisement generation $\mathsf{RepAds}(\mathsf{pp}, \mathsf{urep}[i], \mathsf{scr}[i], m, P)$**:**

– Parse $\mathsf{urep}[i] = (k, s, e, R, C)$ and $\mathsf{scr}[i] = (n_1, n_2, \cdots, n_v, t)$
– Choose $\alpha, r_e, r_t, r_k, r_\alpha, r_\rho, r_{n_1}, \cdots, r_{n_v}, r_t \xleftarrow{\$} \mathbb{Z}_p^*$, $d \xleftarrow{\$} \mathbb{G}_1$. Compute $E = C g_{v+3}^{\alpha}$, $D = d^k$, $\mathsf{aid} = (d, D)$, $R_1 = d^{r_k}$,

$$R_2 = \hat{\mathsf{e}}(E, w)^{r_e} \hat{\mathsf{e}}(g_{v+1}, w)^{r_t} \hat{\mathsf{e}}(g_{v+2}, w)^{r_k} \hat{\mathsf{e}}(g_{v+3}, w)^{r_\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{r_\alpha} \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{r_{n_i}}$$

– Generate a range proof to show $P = \bigwedge_{i=1}^{\zeta} \left(\psi_i \in [0, 2^{\ell_i})\right)$ holds on $\mathsf{scr}[i]$. Compute

$$\mathbf{a}_L \in \{0,1\}^{\ell} \text{ s.t. } \langle \mathbf{a}_{L,[\ell_{i-1},\ell_i]}, \mathbf{2}^{\ell_i} \rangle = \psi_i \text{ for } i \in [1, \zeta],\ \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^{\ell}$$
$$\eta_1, \eta_2 \xleftarrow{\$} \mathbb{Z}_p^*,\ \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^{\ell}, A = h^{\eta_1} \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R},\ S = h^{\eta_2} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$$
$$y_1 = \mathcal{H}(A, S),\ y_2 = \mathcal{H}(A, S, y_1),\ y_3 = \mathcal{H}(A, S, y_1, y_2)$$
$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p^*$$
$$\tau = \tau_1 \cdot y_3 + \tau_2 \cdot y_3^2 + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot r_{\psi_i} \text{ with } r_{\psi_i} = \sum_{j=1}^{v} \beta_{ij} r_{n_j} \text{ for } i \in [1, \zeta - 1] \text{ and } r_{\psi_\zeta} = r_t$$
$$\mathbf{l} = \mathbf{a}_L - y_2 \cdot \mathbf{1}^{\ell} + \mathbf{s}_L \cdot y_3$$
$$\mathbf{r} = \mathbf{y_1}^{\ell} \circ (\mathbf{a}_R + y_2 \cdot \mathbf{1}^{\ell} + \mathbf{s}_R \cdot y_3) + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (\mathbf{0}^{\ell_1 + \cdots + \ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel \mathbf{0}^{\ell_{i+1} + \cdots + \ell_\zeta})$$
$$f = \langle \mathbf{l}, \mathbf{r} \rangle,\ \eta = \eta_1 + \eta_2 \cdot y_3$$

Construct two degree 1 polynomials $l(X)$ and $r(X)$ in $\mathbb{Z}_p^{\ell}[X]$ and compute f_0, f_1, f_2 such that:

$$l(X) = \mathbf{a}_L - y_2 \cdot \mathbf{1}^{\ell} + \mathbf{s}_L \cdot X$$
$$r(X) = \mathbf{y_1}^{\ell} \circ (\mathbf{a}_R + y_2 \cdot \mathbf{1}^{\ell} + \mathbf{s}_R \cdot X) + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (\mathbf{0}^{\ell_1 + \cdots + \ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel \mathbf{0}^{\ell_{i+1} + \cdots + \ell_\zeta})$$
$$f(X) = \langle l(X), r(X) \rangle = f_0 + f_1 X + f_2 X^2$$

– Compute $c = \mathcal{H}(\mathsf{aid}, R_1, R_2, E, A, S, y_1, y_2, y_3, \tau, \eta, f, \mathbf{l}, \mathbf{r}, m)$,

$$z_e = e \cdot c - r_e \quad z_k = k \cdot c + r_k \quad z_t = t \cdot c + r_t \quad z_\rho = (s + \alpha e) \cdot c + r_\rho$$
$$z_{f_1} = f_1 \cdot c + \tau_1 \quad z_{f_2} = f_2 \cdot c + \tau_2 \quad z_\alpha = \alpha \cdot c + r_\alpha \text{ for } j \in [1, v] : z_{n_j} = n_j \cdot c + r_{n_j}$$

– Output $(\mathsf{aid}, \pi_{\mathsf{rep}})$ where $\pi_{\mathsf{rep}} = (E, A, S, \eta, c, z_e, z_t, z_k, z_\alpha, z_\rho, \{z_{n_j}\}_{j=1}^{v}, z_{f_1}, z_{f_2}, \mathbf{l}, \mathbf{r})$

Remark 1. Note that the vectors $\mathbf{l}, \mathbf{r} \in \mathbb{Z}_p^\ell$ contain 2ℓ elements in $\mathbb{Z}_p^*$. The transmission of $\mathbf{l}$ and $\mathbf{r}$ can be replaced with an inner-product proof for statement $f = \langle \mathbf{l}, \mathbf{r} \rangle$ as stated in [14]. The inner-product proof only consists of $2\lceil \log \ell \rceil$ elements in $\mathbb{G}_1$ and 2 elements in $\mathbb{Z}_p$. The proof is only for avoiding transmitting $\mathbf{l}, \mathbf{r}$ and there is no need to keep $\mathbf{l}, \mathbf{r}$ secure. For simplicity, we ignore this optimisation in our discussion.

Advertisement verification $\mathsf{RepVer}(\mathsf{pp}, \mathsf{aid}, \pi_{\mathsf{rep}}, m, P)$**:**

– Parse $\mathsf{aid} = (d, D)$ and $\pi_{\mathsf{rep}} = (E, A, S, \eta, c, z_e, z_t, z_k, z_\alpha, z_\rho, \{z_{n_j}\}_{j=1}^{v}, z_{f_1}, z_{f_2}, \mathbf{l}, \mathbf{r})$.
– Compute

$$\tilde{R}_1 = d^{z_k} D^{-c} \tag{1}$$

$$\tilde{R}_2 = \left(\frac{\hat{\mathsf{e}}(g_0, w)}{\hat{\mathsf{e}}(E, W)}\right)^c \hat{\mathsf{e}}(E, w)^{-z_e}\, \hat{\mathsf{e}}(g_{v+1}, w)^{z_t} \hat{\mathsf{e}}(g_{v+2}, w)^{z_k} \hat{\mathsf{e}}(g_{v+3}, w)^{z_\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{z_\alpha} \cdot \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{z_{n_i}} \tag{2}$$

$$\tilde{f} = \langle \mathbf{l}, \mathbf{r} \rangle \tag{3}$$

$$\tilde{y}_1 = \mathcal{H}(A, S),\ \tilde{y}_2 = \mathcal{H}(A, S, \tilde{y}_1),\ \ \tilde{y}_3 = \mathcal{H}(A, S, \tilde{y}_1, \tilde{y}_2) \tag{4}$$

$$\mu = \tilde{y}_2 \cdot \langle \mathbf{1}^\ell, \tilde{\mathbf{y}}_{\mathbf{1}}^\ell \rangle - \tilde{y}_2^2 \cdot \langle \mathbf{1}^\ell, \tilde{\mathbf{y}}_{\mathbf{1}}^\ell \rangle - \sum_{i=1}^{\zeta} \tilde{y}_2^{i+2} \cdot \langle \mathbf{1}^{\ell_i}, \mathbf{2}^{\ell_i} \rangle \tag{5}$$

$$\tilde{\tau} = \sum_{i=1}^{\zeta-1} \tilde{y}_2^{i+1} (\sum_{j=1}^{v} \beta_{ij} z_{n_j}) + \tilde{y}_2^{\zeta+1}(z_t - \hat{t} \cdot c) + \mu \cdot c + z_{f_1} \cdot y_3 + z_{f_2} \cdot y_3^2 - \tilde{f} \cdot c \tag{6}$$

$$h'_i = h_i^{\tilde{y}_1^{-i+1}} \text{ for } i \in [1, \ell] \tag{7}$$

$$AS^{\tilde{y}_3} \mathbf{g}^{-\tilde{y}_2} \mathbf{h}'^{\tilde{y}_2 \cdot \tilde{\mathbf{y}}_{\mathbf{1}}^\ell + \sum_{i=1}^{\zeta} \tilde{y}_2^{i+1} (\mathbf{0}^{\ell_1 + \cdots + \ell_{i-1}} \| \mathbf{2}^{\ell_i} \| \mathbf{0}^{\ell_{i+1} + \cdots + \ell_\zeta})} \overset{?}{=} h^\eta \mathbf{g}^{\mathbf{l}} \mathbf{h}'^{\mathbf{r}} \tag{8}$$

$$\tilde{c} = \mathcal{H}(\tilde{R}_1, \tilde{R}_2, E, A, S, \tilde{y}_1, \tilde{y}_2, \tilde{y}_3, \tilde{\tau}, \eta, \tilde{f}, \mathbf{l}, \mathbf{r}, m) \tag{9}$$

– If $c = \tilde{c}$ then output 1 otherwise output 0.

Remark 2. We can optimise RepAds to be pairing-free and reduce the number of pairing operations in RepVer. For RepAds, the computation of R_2 can be transformed as

$$\begin{aligned} R_2 &= \hat{\mathsf{e}}(E, w)^{r_e} \hat{\mathsf{e}}(g_{v+1}, w)^{r_t} \hat{\mathsf{e}}(g_{v+2}, w)^{r_k} \hat{\mathsf{e}}(g_{v+3}, w)^{r_\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{r_\alpha} \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{r_{n_i}} \\ &= \hat{\mathsf{e}}(C, w)^{r_e} \hat{\mathsf{e}}(g_{v+1}, w)^{r_t} \hat{\mathsf{e}}(g_{v+2}, w)^{r_k} \hat{\mathsf{e}}(g_{v+3}, w)^{\alpha r_e + r_\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{r_\alpha} \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{r_{n_i}} \end{aligned}$$

The pairings $\hat{\mathsf{e}}(C, w)$, $\hat{\mathsf{e}}(g_{v+3}, W)$ and $\{\hat{\mathsf{e}}(g_i, w)\}_{i=1}^{v+3}$ can be pre-computed and are reusable since they do not depend on any variables generated in the advertising phase. For RepVer, the number of pairing operations can be reduced to two by modifying

$$\begin{aligned} \tilde{R}_2 &= (\hat{\mathsf{e}}(g_0, w)/\hat{\mathsf{e}}(E, W))^c \hat{\mathsf{e}}(E, w)^{-z_e}\, \hat{\mathsf{e}}(g_{v+1}, w)^{z_t} \hat{\mathsf{e}}(g_{v+2}, w)^{z_k} \hat{\mathsf{e}}(g_{v+3}, w)^{z_\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{z_\alpha} \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{z_{n_i}} \\ &= \hat{\mathsf{e}}(g_0^c E^{-z_e} g_{v+1}^{z_t} g_{v+2}^{z_k} g_{v+3}^{z_\rho} g_i^{\sum_{i=1}^{v} z_{n_i}}, w)\, \hat{\mathsf{e}}(E^{-c} g_{v+3}^{z_\alpha}, W) \end{aligned}$$

B.2 Security analysis of advertising protocol

In the following we prove that our advertising protocol $(\mathsf{RepAds}, \mathsf{RepVer})$ fulfills the properties of completeness, zero-knowledge and proof-of-knowledge.

Lemma 1. *The advertising protocol is complete.*

Proof. When a proof is generated honestly following the specification of the protocol, we shall verify that $\tilde{R}_1 = R_1$, $\tilde{R}_2 = R_2$, $\tilde{f} = f$, $\tilde{y}_1 = y_1$, $\tilde{y}_2 = y_2$, $\tilde{y}_3 = y_3$, and $\tilde{\tau} = \tau$.

$$\begin{aligned}
\tilde{R}_1 &= \left(\frac{\hat{\mathsf{e}}(g_0, w)}{\hat{\mathsf{e}}(E, W)}\right)^c \hat{\mathsf{e}}(E, w)^{-z_e} \hat{\mathsf{e}}(g_{v+1}, w)^{z_t} \hat{\mathsf{e}}(g_{v+2}, w)^{z_k} \hat{\mathsf{e}}(g_{v+3}, w)^{z_\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{z_\alpha} \cdot \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{z_{n_i}} \\
&= \left(\frac{\hat{\mathsf{e}}(g_0, w)}{\hat{\mathsf{e}}(E, W)}\right)^c \hat{\mathsf{e}}(E, w)^{-e\cdot c + r_e} \hat{\mathsf{e}}(g_{v+1}, w)^{t\cdot c + r_t} \hat{\mathsf{e}}(g_{v+2}, w)^{k\cdot c + r_k} \hat{\mathsf{e}}(g_{v+3}, w)^{\rho\cdot c + r_\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{\alpha\cdot c + r_\alpha} \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{n_i\cdot c + r_{n_i}} \\
&= R_1 \cdot \left(\frac{\hat{\mathsf{e}}(g_0, w)}{\hat{\mathsf{e}}(E, W)}\right)^c \hat{\mathsf{e}}(E, w)^{-e\cdot c} \hat{\mathsf{e}}(g_{v+1}, w)^{t\cdot c} \hat{\mathsf{e}}(g_{v+2}, w)^{k\cdot c} \hat{\mathsf{e}}(g_{v+3}, w)^{\rho} \hat{\mathsf{e}}(g_{v+3}, W)^{\alpha\cdot c} \cdot \prod_{i=1}^{v} \hat{\mathsf{e}}(g_i, w)^{n_i\cdot c} \\
&= R_1 \cdot \left(\frac{\hat{\mathsf{e}}(g_0 g_1^{n_1} \cdots g_v^{n_v} g_{v+1}^{t} g_{v+2}^{k} g_{v+3}^{s+(\gamma+e)\alpha}, w)}{\hat{\mathsf{e}}(E^{\gamma+e}, w)}\right)^c = R_1 \\
\tilde{R}_2 &= d^{z_k} D^{-c} = d^{k\cdot c + r_k} D^{-c} = \frac{d^{k\cdot c}}{D^c} R_2 = R_2
\end{aligned}$$

It is easy to see that $\tilde{f} = f, \tilde{y}_1 = y_1, \tilde{y}_2 = y_2$ and $\tilde{y}_3 = y_3$. To verify $\tilde{\tau} = \tau$, we first compute the value of f_0. Since

$$\langle \mathbf{l}, \mathbf{r} \rangle = \left\langle \mathbf{a}_L - y_2 \cdot \mathbf{1}^\ell + \mathbf{s}_L \cdot y_3,\ \ \mathbf{y_1}^\ell \circ (\mathbf{a}_R + y_2 \cdot \mathbf{1}^\ell + \mathbf{s}_R \cdot y_3) + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel 0^{\ell_{i+1}+\cdots+\ell_\zeta}\ ) \right\rangle$$

we can compute f_0 as

$$\begin{aligned}
f_0 &= \langle \mathbf{a}_L - y_2 \cdot \mathbf{1}^\ell,\ \mathbf{y_1}^\ell \circ (\mathbf{a}_R + y_2 \cdot \mathbf{1}^\ell) + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel 0^{\ell_{i+1}+\cdots+\ell_\zeta}) \rangle \\
&= \langle \mathbf{a}_L, \mathbf{y_1}^\ell \circ (\mathbf{a}_R + y_2 \cdot \mathbf{1}^\ell) \rangle + \left\langle \mathbf{a}_L, \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel 0^{\ell_{i+1}+\cdots+\ell_\zeta}) \right\rangle \\
&\quad - \langle y_2 \cdot \mathbf{1}^\ell, \mathbf{y_1}^\ell \circ (\mathbf{a}_R + y_2 \cdot \mathbf{1}^\ell) \rangle - \left\langle y_2 \cdot \mathbf{1}^\ell, \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel 0^{\ell_{i+1}+\cdots+\ell_\zeta}) \right\rangle \\
&= \langle \mathbf{a}_L, \mathbf{y_1}^\ell \circ \mathbf{a}_R \rangle + \langle \mathbf{a}_L - \mathbf{a}_R, y_2 \cdot \mathbf{y_1}^\ell \rangle + \sum_{i=1}^{\zeta} y_2^{i+1} \psi_i - y_2^2 \cdot \langle \mathbf{1}^\ell, \mathbf{y_1}^\ell \rangle - \sum_{i=1}^{\zeta} y_2^{i+2} \cdot \langle \mathbf{1}^{\ell_i}, \mathbf{2}^{\ell_i} \rangle \\
&= (\sum_{i=1}^{\zeta} y_2^{i+1} \psi_i) + y_2 \cdot \langle \mathbf{1}^\ell, \mathbf{y_1}^\ell \rangle - y_2^2 \cdot \langle \mathbf{1}^\ell, \mathbf{y_1}^\ell \rangle - \sum_{i=1}^{\zeta} y_2^{i+2} \cdot \langle \mathbf{1}^{\ell_i}, \mathbf{2}^{\ell_i} \rangle \\
&= (\sum_{i=1}^{\zeta} y_2^{i+1} \psi_i) + \mu
\end{aligned}$$

Therefore

$$\begin{aligned}
\tilde{\tau} &= \sum_{i=1}^{\zeta-1} y_2^{i+1} (\sum_{j=1}^{v} \beta_{ij} z_{n_j}) + y_2^{\zeta+1} (z_t - \hat{t} \cdot c) + \mu \cdot c + z_{f_1} \cdot y_3 + z_{f_2} \cdot y_3^2 - f \cdot c \\
&= \sum_{i=1}^{\zeta-1} y_2^{i+1} (\sum_{j=1}^{v} \beta_{ij} (n_j \cdot c + r_{n_j})) + y_2^{\zeta+1} (t \cdot c + r_t - \hat{t} \cdot c) + \mu \cdot c + (f_1 \cdot c + \tau_1) \cdot y_3 + (f_2 \cdot c + \tau_2) \cdot y_3^2 - f \cdot c \\
&= \sum_{i=1}^{\zeta-1} (y_2^{i+1} \cdot \psi_i \cdot c) + \sum_{i=1}^{\zeta-1} y_2^{i+1} (\sum_{j=1}^{v} \beta_{ij} r_{n_j}) + y_2^{\zeta+1} \cdot \psi_\zeta \cdot c \\
&\qquad + y_2^{\zeta+1} \cdot r_\zeta + \mu \cdot c + f_1 \cdot c \cdot x + f_2 \cdot c \cdot y_3^2 - f \cdot c + \tau_1 \cdot y_3 + \tau_2 \cdot y_3^2
\end{aligned}$$

$$= \sum_{i=1}^{\zeta}(y_2^{i+1} \cdot \psi_i \cdot c) + \mu \cdot c + f_1 \cdot c \cdot y_3 + f_2 \cdot c \cdot y_3^2 - f \cdot c + \tau_1 \cdot y_3 + \tau_2 \cdot y_3^2 + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot r_{\psi_i}$$
$$= f_0 \cdot c + f_1 \cdot c \cdot y_3 + f_2 \cdot c \cdot y_3^2 - f \cdot c + \tau = \tau$$

Equation 8 holds because:

$$\begin{aligned}
&AS^{y_3} \cdot \mathbf{g}^{-y_2} \cdot \mathbf{h}'^{y_2 \cdot \mathbf{y_1}^{\ell} + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \| \mathbf{2}^{\ell_i} \| 0^{\ell_{i+1}+\cdots+\ell_\zeta})} \\
&= h^{\eta_1}\mathbf{g}^{\mathbf{a}_L}\mathbf{h}^{\mathbf{a}_R}(h^{\eta_2}\mathbf{g}^{\mathbf{s}_L}\mathbf{h}^{\mathbf{s}_R})^{y_3} \cdot \mathbf{g}^{-y_2} \cdot \mathbf{h}'^{y_2 \cdot \mathbf{y_1}^{\ell} + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \| \mathbf{2}^{\ell_i} \| 0^{\ell_{i+1}+\cdots+\ell_\zeta})} \\
&= h^{\eta_1+\eta_2 \cdot y_3}\mathbf{g}^{\mathbf{a}_L + \mathbf{s}_L \cdot y_3 - y_2 \cdot \mathbf{1}^{\ell}} \cdot \mathbf{h}'^{(\mathbf{a}_R + \mathbf{s}_R \cdot y_3 + y_2 \cdot \mathbf{1}^{\ell}) \circ \mathbf{y_1}^{\ell} + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (\mathbf{0}^{\ell_1+\cdots+\ell_{i-1}} \| \mathbf{2}^{\ell_i} \| \mathbf{0}^{\ell_{i+1}+\cdots+\ell_\zeta})} \\
&= h^{\eta}\mathbf{g}^{\mathbf{l}}\mathbf{h}'^{\mathbf{r}}
\end{aligned}$$

Lemma 2. *The advertising protocol can be simulated.*

Proof. We describe how to simulate $(\mathsf{aid}, \pi_{\mathsf{rep}})$. Choose all the elements and challenges in the proof uniformly at random, i.e., $d, D, E, A \xleftarrow{\$} \mathbb{G}_1$ and $y_1, y_2, y_3, \eta, c, z_e, z_t, z_k, z_\alpha, z_\rho, z_{f_1}, z_{f_2} \xleftarrow{\$} \mathbb{Z}_p^*$ and $z_{n_j} \xleftarrow{\$} \mathbb{Z}_p^*$ for $j \in [1, v]$ and $\mathbf{l}, \mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^{\ell}$. Compute $f = \langle \mathbf{l}, \mathbf{r} \rangle$ and R_1, R_2, τ according to the verification equations and

$$S = (h^{\eta}\mathbf{g}^{\mathbf{l}}\mathbf{h}'^{\mathbf{r}}(A \cdot \mathbf{g}^{-y_2} \cdot \mathbf{h}'^{y_2 \cdot \mathbf{y_1}^{\ell}} \prod_{i=1}^{\zeta} \mathbf{h}'^{y_2^{i+1} \cdot \mathbf{2}_i^{\ell}}_{[\ell_1+\cdots+\ell_{i-1}, \ell_1+\cdots+\ell_i]})^{-1})^{y_3^{-1}}$$

The output is $\mathsf{aid} = (d, D)$ and $\pi_{\mathsf{rep}} = (E, A, S, \eta, c, z_e, z_t, z_k, z_\alpha, z_\rho, \{z_{n_j}\}_{j\in[1,v]}, z_{f_1}, z_{f_2}, f, \mathbf{l}, \mathbf{r})$. The simulated $(\mathsf{aid}, \pi_{\mathsf{rep}})$ is distributed identically to the real proof.

Lemma 3. *There exists an extractor for the advertising protocol.*

Proof. Suppose that an extractor can rewind a prover to the point just before the challenge c. To challenge c', the prover generates $z'_e, z'_t, z'_k, z'_\alpha, z'_\rho, \{z'_{n_j}\}_{j\in[1,v]}, z'_{f_1}, z'_{f_2}$. Let $\Delta e = \frac{z_e - z'_e}{c - c'}$, and similarly for $\Delta t, \Delta k, \Delta\alpha, \Delta\rho, \{\Delta n_j\}_{j\in[1,v]}$, $\Delta f_1, \Delta f_2$. Let $\Delta s = \Delta\rho - \Delta\alpha\Delta e$.

– Consider (2) in the verification equation. Dividing the two instances of this equation, we obtain

$$\left(\frac{\hat{\mathsf{e}}(g_0, w)}{\hat{\mathsf{e}}(E, W)}\right)\hat{\mathsf{e}}(E, w)^{-\Delta e}\hat{\mathsf{e}}(g_{v+1}, w)^{\Delta t}\hat{\mathsf{e}}(g_{v+2}, w)^{\Delta k}\hat{\mathsf{e}}(g_{v+3}, w)^{\Delta\rho}\hat{\mathsf{e}}(g_{v+3}, W)^{\Delta\alpha}\prod_{i=1}^{v}\hat{\mathsf{e}}(g_i, w)^{\Delta n_i} = 1$$

This can be rearranged as

$$\begin{aligned}
\hat{\mathsf{e}}(E, W)\hat{\mathsf{e}}(E, w)^{\Delta e} &= \hat{\mathsf{e}}(g_0, w)\hat{\mathsf{e}}(g_{v+1}, w)^{\Delta t}\,\hat{\mathsf{e}}(g_{v+2}, w)^{\Delta k}\,\hat{\mathsf{e}}(g_{v+3}, w)^{\Delta\rho}\,\hat{\mathsf{e}}(g_{v+3}, W)^{\Delta\alpha}\,\prod_{j=1}^{v}\hat{\mathsf{e}}(g_j, w)^{\Delta n_j} \\
&= \hat{\mathsf{e}}(g_0, w)\hat{\mathsf{e}}(g_{v+1}, w)^{\Delta t}\,\hat{\mathsf{e}}(g_{v+2}, w)^{\Delta k}\,\hat{\mathsf{e}}(g_{v+3}, w)^{\Delta s + \Delta\alpha\Delta e}\,\hat{\mathsf{e}}(g_{v+3}, w)^{\gamma\Delta\alpha}\,\prod_{j=1}^{v}\hat{\mathsf{e}}(g_j, w)^{\Delta n_j} \\
&= \hat{\mathsf{e}}\left(\left(g_0 g_1^{\Delta n_1} \cdots g_v^{\Delta n_v} g_{v+1}^{\Delta t} g_{v+2}^{\Delta k} g_{v+3}^{\Delta s}\right)^{\frac{1}{\gamma + \Delta e}} g_{v+3}^{\Delta\alpha}, w\right)^{\gamma + \Delta e}
\end{aligned}$$

Since $\hat{\mathsf{e}}(E, W)\hat{\mathsf{e}}(E, w)^{\Delta e} = \hat{\mathsf{e}}(E, w)^{\gamma + \Delta e}$, we can deduce that

$$\hat{\mathsf{e}}(E, w) = \hat{\mathsf{e}}\left(\left(g_0 g_1^{\Delta n_1} \cdots g_v^{\Delta n_v} g_{v+1}^{\Delta t} g_{v+2}^{\Delta k} g_{v+3}^{\Delta s}\right)^{\frac{1}{\gamma + \Delta e}} g_{v+3}^{\Delta\alpha}, w\right)$$
$$\hat{\mathsf{e}}(E g_{v+3}^{-\Delta\alpha}, w) = \hat{\mathsf{e}}\left(\left(g_0 g_1^{\Delta n_1} \cdots g_v^{\Delta n_v} g_{v+1}^{\Delta t} g_{v+2}^{\Delta k} g_{v+3}^{\Delta s}\right)^{\frac{1}{\gamma + \Delta e}}, w\right)$$

Let $C^* = Eg_{v+3}^{-\Delta\alpha} = \left(g_0 g_1^{\Delta n_1} \cdots g_v^{\Delta n_v} g_{v+1}^{\Delta t} g_{v+2}^{\Delta k} g_{v+3}^{\Delta s}\right)^{\frac{1}{\gamma+\Delta e}}$. The algorithm extracts the tuple $(C^*, \Delta n_1, \cdots, \Delta n_v, \Delta t, \Delta k, \Delta s, \Delta e)$.

- Consider (1) in the verification equation. Dividing the two instances of this equation, we obtain $1 = d^{z_k - z'_k} D^{-c+c'}$ and therefore $D = d^{\Delta k}$.
- Assume two valid proofs with different challenges y_3, y'_3. To challenge y'_3, the extractor generates $\mathbf{l}', \mathbf{r}', \eta'$. Let $\Delta\eta_2 = \frac{\eta-\eta'}{y_3-y'_3}$, $\Delta\mathbf{s}_L = \frac{\mathbf{l}-\mathbf{l}'}{y_3-y'_3}$, $\Delta\mathbf{s}_R = \frac{(\mathbf{r}-\mathbf{r}')\circ\mathbf{y_1}^{\ell}}{y_3-y'_3}$, $\Delta\eta_1 = \eta - \Delta\eta_2 \cdot y_3$, $\Delta\mathbf{a}_L = \mathbf{l} + y_2 \cdot \mathbf{1}^{\ell} - \Delta\mathbf{s}_L \cdot y_3$ and $\Delta\mathbf{a}_R = \mathbf{r} \circ \mathbf{y_1}^{-\ell} - y_2 \cdot \mathbf{1}^{\ell} - \Delta\mathbf{s}_R \cdot y_3 - \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel 0^{\ell_{i+1}+\cdots+\ell_\zeta}) \circ \mathbf{y_1}^{-\ell}$. Consider (8) in the verification equation and divide the two instances of this equation, we obtain $S^{y_3-y'_3} = h^{\eta-\eta'} \mathbf{g}^{\mathbf{l}-\mathbf{l}'} \mathbf{h}'^{\mathbf{r}-\mathbf{r}'}$ which means

$$S = h^{\frac{\eta-\eta'}{y_3-y'_3}} \mathbf{g}^{\frac{\mathbf{l}-\mathbf{l}'}{y_3-y'_3}} \mathbf{h}^{\frac{(\mathbf{r}-\mathbf{r}')\circ\mathbf{y_1}^{\ell}}{y_3-y'_3}} = h^{\Delta\eta_2} \mathbf{g}^{\Delta\mathbf{s}_L} \mathbf{h}^{\Delta\mathbf{s}_R}$$

  Using (8) to compute A

$$\begin{aligned} A &= \frac{h^{\eta} \mathbf{g}^{\mathbf{l}} \mathbf{h}'^{\mathbf{r}}}{S^{y_3} \cdot \mathbf{g}^{-y_2} \cdot \mathbf{h}'^{y_2 \cdot \mathbf{y_1}^{\ell}} \cdot \prod_{i=1}^{\zeta} \mathbf{h}'^{\tilde{y}_2^{i+1} \cdot \mathbf{2}_i^{\ell}}_{[\ell_1+\cdots+\ell_{i-1}, \ell_1+\cdots+\ell_i]}} \\ &= h^{\eta-\Delta\eta_2 \cdot y_3} \mathbf{g}^{\mathbf{l}+y_2\cdot\mathbf{1}^{\ell}-\Delta\mathbf{s}_L\cdot y_3} \mathbf{h}^{\mathbf{r}\circ\mathbf{y_1}^{-\ell}-y_2\cdot\mathbf{1}^{\ell}-\Delta\mathbf{s}_R\cdot y_3-\sum_{i=1}^{\zeta}\tilde{y}_3^{i+1}\cdot(\mathbf{0}^{\ell_1+\cdots+\ell_{i-1}}\|\mathbf{2}^{\ell_i}\|\mathbf{0}^{\ell_{i+1}+\cdots+\ell_\zeta})\circ\mathbf{y_1}^{-\ell}} \\ &= h^{\Delta\eta_1} \mathbf{g}^{\Delta\mathbf{a}_L} \mathbf{h}^{\Delta\mathbf{a}_R} \end{aligned}$$

  and

$$\begin{aligned} \mathbf{l} &= \Delta\mathbf{a}_L + \Delta\mathbf{s}_L \cdot y_3 - y_2 \cdot \mathbf{1}^{\ell} \\ \mathbf{r} &= (\Delta\mathbf{a}_R + y_2 \cdot \mathbf{1}^{\ell} + \Delta\mathbf{s}_R \cdot y_3) \circ \mathbf{y_1}^{\ell} + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel 0^{\ell_{i+1}+\cdots+\ell_\zeta}) \end{aligned}$$

  The values of $\Delta\mathbf{a}_L, \Delta\mathbf{a}_R, \Delta\mathbf{s}_L, \Delta\mathbf{s}_R$ are unique and the equalities hold for all challenges and $\mathbf{l}, \mathbf{r}$ from the transcript, otherwise we have two distinct representations of the same group element using a set of independent generators which breaks the discrete logarithm assumption.
- Let $F(X) = \langle \mathbf{l}(X), \mathbf{r}(X) \rangle = F_0 + F_1 X + F_2 X^2$ and $f = \langle \mathbf{l}, \mathbf{r} \rangle$. Using two different challenges c, c' and subtracting two instances of (6), we obtain

$$\begin{aligned} 0 = &\sum_{i=1}^{\zeta-1} \mu^{i+1} (\sum_{j=1}^{v} \beta_{ij} (z_{n_j} - z'_{n_j})) + y_2^{\zeta+1} (z_t - z'_t - \hat{t} \cdot (c - c')) \\ &\qquad + \mu \cdot (c - c') + (z_{f_1} - z'_{f_1}) \cdot y_3 + (z_{f_2} - z'_{f_2}) \cdot y_3^2 - f \cdot (c - c') \\ \Longrightarrow& \\ f = &\sum_{i=1}^{\zeta-1} \mu^{i+1} (\sum_{j=1}^{v} \beta_{ij} \Delta n_j) + y_2^{\zeta+1} (\Delta t - \hat{t}) + \mu + \Delta f_1 \cdot y_3 + \Delta f_2 \cdot y_3^2 \end{aligned}$$

  Using 3 different values of y_3, we can see that $F_0 = \sum_{i=1}^{\zeta-1} y_2^{i+1} (\sum_{j=1}^{v} \beta_{ij} \Delta n_j) + y_2^{\zeta+1} (\Delta t - \hat{t}) + \mu$. Therefore,

$$\begin{aligned} &\sum_{i=1}^{\zeta-1} y_2^{i+1} (\sum_{j=1}^{v} \beta_{ij} \Delta n_j) + y_2^{\zeta+1} (\Delta t - \hat{t}) + \mu \\ &= \left\langle \Delta\mathbf{a}_L - y_2 \cdot \mathbf{1}^{\ell},\ \mathbf{y_1}^{\ell} \circ (\Delta\mathbf{a}_R + y_2 \cdot \mathbf{1}^{\ell}) + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot (0^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel 0^{\ell_{i+1}+\cdots+\ell_\zeta}) \right\rangle \\ &= \langle \Delta\mathbf{a}_L,\ \mathbf{y_1}^{\ell} \circ \Delta\mathbf{a}_R \rangle + y_2 \cdot \langle \Delta\mathbf{a}_L - \Delta\mathbf{a}_R, \mathbf{y_1}^{\ell} \rangle \\ &\quad + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot \langle \Delta\mathbf{a}_L, (\mathbf{0}^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel \mathbf{0}^{\ell_{i+1}+\cdots+\ell_\zeta}) \rangle + \mu - y_2 \cdot \langle \mathbf{1}^{\ell}, \mathbf{y_1}^{\ell} \rangle \end{aligned}$$

Removing μ from both sides of the equation, we obtain

$$\sum_{i=1}^{\zeta-1} y_2^{i+1}(\sum_{j=1}^{v}\beta_{ij}\Delta n_j) + y_2^{\zeta+1}(\Delta t - \hat{t}) = \langle \Delta\mathbf{a}_L, \mathbf{y_1}^{\ell} \circ \Delta\mathbf{a}_R\rangle +$$

$$y_2 \cdot \langle \Delta\mathbf{a}_L - \Delta\mathbf{a}_R - \mathbf{1}^{\ell}, \mathbf{y_1}^{\ell}\rangle + \sum_{i=1}^{\zeta} y_2^{i+1} \cdot \langle \Delta\mathbf{a}_L, (\mathbf{0}^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel \mathbf{0}^{\ell_{i+1}+\cdots+\ell_\zeta})\rangle$$

Using ℓ y_1 challenges and 4 y_2 challenges, we can infer the following:

$$\Delta\mathbf{a}_L \circ \Delta\mathbf{a}_R = \mathbf{0}^{\ell} \qquad \Delta\mathbf{a}_R = \Delta\mathbf{a}_L - \mathbf{1}^{\ell}$$

$$\sum_{j=1}^{v}\beta_{ij}\Delta n_j = \langle \Delta\mathbf{a}_L, (\mathbf{0}^{\ell_1+\cdots+\ell_{i-1}} \parallel \mathbf{2}^{\ell_i} \parallel \mathbf{0}^{\ell_{i+1}+\cdots+\ell_\zeta})\rangle \text{ for } i = 1, \cdots, \zeta-1$$

$$\Delta t - \hat{t} = \langle \Delta\mathbf{a}_L, \mathbf{0}^{\ell_1+\cdots+\ell_{\zeta-1}} \parallel \mathbf{2}^{\ell_\zeta}\rangle$$

The first two equations imply that $\Delta\mathbf{a}_L \in \{0,1\}^{\ell}$. The last two equations imply that $\sum_{j=1}^{v}\beta_{ij}\Delta n_j \in [0, 2^{\ell_i}]$ and $\Delta t - \hat{t} \in [0, 2^{\ell_\zeta}]$.

C Formal proofs of correctness and security

Theorem 6. *The pRate scheme is correct.*

Proof. RepVer is correct because of Lemma 1. RateAcc is correct because $c_2/c_1^{\xi} = z^{k_0} \cdot U^{a_0}/(u^{a_0})^{\xi} = z^{k_0} = Z_{i_0}$ and $c_2'/{c_1'}^{\xi} = z^{k_1} \cdot U^{a_1}/(u^{a_1})^{\xi} = z^{k_1} = Z_{i_1}$. Upd is correct because $\hat{\mathsf{e}}(C', W \cdot w^{e'}) = \hat{\mathsf{e}}((R \cdot g_{v+1}^{\tilde{t}-t} \cdot V \cdot g_{v+3}^{\tilde{s}})^{\frac{1}{\gamma+e'}}, w^{\gamma} \cdot w^{e'}) = \hat{\mathsf{e}}(R \cdot g_{v+1}^{\tilde{t}-t} \cdot V \cdot g_{v+3}^{\tilde{s}}, w)$.

Theorem 1. *The pRate scheme is anonymous under the XDH assumption.*

Proof. The proof proceeds with a sequence of games.

Game 0. Let Game 0 be the original game $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{anony}}(1^{\lambda})$. Let S_0 be the event that $b = \hat{b}$ in this game. Clearly we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anony}}(1^{\lambda}) = |\mathsf{Pr}[S_0] - \frac{1}{2}|$.

Game 1. Let Game 1 be the same as Game 0, except $\mathsf{ct}_2 \xleftarrow{\$} \mathbb{G}_1$ for user i_b in the challenge σ^* is chosen uniformly at random and the corresponding proofs $\pi_{\mathsf{enc}}, \pi_{\mathsf{tok}}, \pi_{\mathsf{sub}}$ that involve the ct_2 are simulated. For updating queries RateUpd which involve verification and decryption of such simulated proofs $\pi_{\mathsf{tok}}, \pi_{\mathsf{sub}}$, $\mathcal{B}$ maps the result of decryption to user i_b's identity K_{i_b}. Let S_1 be the event that $b = \hat{b}$ in this game. We shall prove that $|\mathsf{Pr}[S_1] - \mathsf{Pr}[S_0]| \le \mathsf{negl}(\lambda)$ under the XDH assumption in the random oracle model. Assume an adversary $\mathcal{A}$ breaks the anonymity of the pRate scheme. We shall construct an adversary $\mathcal{B}$ which breaks the XDH assumption.
Let $(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}})$ be an XDH problem given to $\mathcal{B}$ where $\mathbb{G}_1$ is assumed to be associated with a bilinear map $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathcal{B}$'s goal is to decide whether $\mathcal{Z} = \mathcal{X}\mathcal{Y}$. $\mathcal{B}$ generates parameters for pRate scheme as follows: $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$, $g_0, g_1, \cdots, g_{v+3}, g, u \xleftarrow{\$} \mathbb{G}_1$, $w \xleftarrow{\$} \mathbb{G}_2$. Compute $W = w^{\gamma}$. Let $u = g'$ and $U = {g'}^{\mathcal{X}}$. The master issuing key is $\mathsf{ik} = \gamma$. The master opening key is $\mathsf{ok} = \mathcal{X}$ that is unknown to $\mathcal{B}$. The public parameters are $\mathsf{pp} = (g_0, g_1, \cdots, g_{v+3}, g, w, W, u, U)$. $\mathcal{B}$ gives pp and ik to $\mathcal{A}$. $\mathcal{B}$ can easily answer queries $\mathsf{AddU}, \mathsf{WReg}, \mathsf{CrptU}, \mathsf{RepAds}, \mathsf{Token}, \mathsf{SndTok}, \mathsf{RateGen}$. Below we describe how $\mathcal{B}$ answers hash oracle queries and creates simulated proofs $\pi_{\mathsf{enc}}, \pi_{\mathsf{tok}}, \pi_{\mathsf{sub}}$ in the challenge σ^* and answers RateUpd queries.

- On a hash query $m \in \{0,1\}^*$ to $\mathcal{H}$, if m has not been queried before, then choose $c \xleftarrow{\$} \mathbb{Z}_p^*$ and set $\mathcal{H}(m) = c$. Return $\mathcal{H}(m)$ to $\mathcal{A}$.
- The challenge $\sigma^* = (\mathsf{aid}^*, \pi_{\mathsf{rep}}^*, \mathsf{sn}', \mathsf{RT}', \mathsf{UT}', \delta^*)$ is constructed in the same way as in Game 0 except that $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2) = (g^{\mathcal{Y}}, K_{i_b} \cdot g^{\mathcal{Z}})$ in RT and RT' and δ^*. Since $\mathcal{B}$ does not know the value of $\mathcal{Y}$, $\mathcal{B}$ simulates π_{enc}, π_{tok} and π_{sub} as below:

- To simulate π_{enc}, choose $c, \varrho_a, \varrho_k \xleftarrow{\$} \mathbb{Z}_p^*$, compute R_1, R_2, R_3 according to the verification equations in Figure 2 and set $\mathcal{H}(\mathsf{ct}, D, R_1, R_2, R_3) = c$.
- To simulate π_{tok}, choose $c, \varrho_a, \varrho_k \xleftarrow{\$} \mathbb{Z}_p^*$, let $Z = g^{\mathcal{Z}}$, $R_1 = u^{\varrho_a}/\mathsf{ct}_1^c$, $R_2 = U^{\varrho_a}/(g^{\mathcal{Z}})^c$, $R_3 = g_{v+2}^{\varrho_k} U^{\varrho_a} R_2^{-1} \mathsf{ct}_2^{-c}$ and $\mathcal{H}(\mathsf{ct}, Z, R_1, R_2, R_3, \mathsf{sn}, \mathsf{sn}', \mathsf{ct}') = c$. The proof $\pi_{\mathsf{tok}} = (\mathsf{ct}, c, \varrho_a, \varrho_k)$.
- To simulate π_{sub}, choose $\varrho_a, \varrho_k \xleftarrow{\$} \mathbb{Z}_p^*$ and $e_j, \varrho_j \xleftarrow{\$} \mathbb{Z}_p^*$ for $j \in [1, v]$. Let $Z = g^{\mathcal{Z}}$, $c = \sum_{j=1}^{v} e_j$, $A_j = g_{v+3}^{\varrho_j}(g_j/V)^{e_j}$, $B_j = g^{\varrho_j}/\mathsf{sn}^{e_j}$ for $j \in [1, v]$ and $R_1 = u^{\varrho_a}/\mathsf{ct}_1^c$, $R_2 = U^{\varrho_a}/(g^{\mathcal{Z}})^c$, $R_3 = g_{v+2}^{\varrho_k} U^{\varrho_a} R_2^{-1} \mathsf{ct}_2^{-c}$ and $\mathcal{H}(\mathsf{ct}, Z, R_1, R_2, R_3, A_1, \cdots, A_v, B_1, \cdots, B_v) = c$. The proof $\pi_{\mathsf{sub}} = (\mathsf{ct}, \varrho_a, \varrho_k, \{e_j, \varrho_j\}_{j=1}^{v})$.

– On a $\mathsf{RateUpd}(\mathsf{sn}, \delta, M_{\mathsf{in}})$ query, if sn has been used before then return $\perp$. Otherwise, $\mathcal{B}$ has to simulate the checking of proofs in δ since $\mathcal{B}$ does not have the master opening key ok. To check if $\delta = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$ is valid, $\mathcal{B}$ checks the validity of π'_{tok} and π_{sub} as below:
- To check π'_{tok}: parse $\pi'_{\mathsf{tok}} = (\mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2), c, \varrho_a, \varrho_k)$, look throughout $\mathcal{H}$ for a query $(\mathsf{ct}', Z, R_1, R_2, R_3, \mathsf{sn}', \mathsf{sn}, \mathsf{ct})$ that maps to c. If there is no such query, it returns $\perp$. $\mathcal{B}$ checks if $R_1 \overset{?}{=} u^{\varrho_a} {\mathsf{ct}'_1}^{-c}$, $R_3 \overset{?}{=} g_{v+2}^{\varrho_k} U^{\varrho_a} R_2^{-1} {\mathsf{ct}_2}^{-c}$ and $U^{\varrho_a} \overset{?}{=} R_2 Z^c$. If the check fails, $\mathcal{B}$ returns $\perp$. Next we prove $Z = \mathsf{ct}_1^{\mathcal{X}}$ and $R_2 = R_1^{\mathcal{X}}$ with overwhelming probability. Assume $Z = \mathsf{ct}_1^{\xi_1}$ and $R_2 = R_1^{\xi_2}$ with $\xi_1 \neq \mathcal{X}$ or $\xi_2 \neq \mathcal{X}$. Assume $R_1 = u^{r_a}$ and $\mathsf{ct}'_1 = u^a$ for some r_a and a. From $R_1 = u^{\varrho_a} {\mathsf{ct}'_1}^{-c}$, we have $u^{r_a} = u^{\varrho_a} u^{-c \cdot a}$ which means $r_a = \varrho_a - c \cdot a$. From $U^{\varrho_a} = R_2 Z^c$, we have $u^{\mathcal{X} \cdot (r_a + c \cdot a)} = u^{r_a \cdot \xi_2} u^{a \cdot \xi_1 \cdot c}$ which gives us $\mathcal{X} \cdot (r_a + c \cdot a) = r_a \cdot \xi_2 + a \cdot \xi_1 \cdot c$. This implies $(\mathcal{X} - \xi_2) r_a = (\xi_1 - \mathcal{X}) ac$. If $\xi_1 = \mathcal{X}$ then $\xi_2 = \mathcal{X}$ and vice versa, which contradicts the assumption that $\xi_1 \neq \mathcal{X}$ or $\xi_2 \neq \mathcal{X}$. Hence we know $\xi_1 \neq \mathcal{X}$ and $\xi_2 \neq \mathcal{X}$. This gives us $c = (\mathcal{X} - \xi_2) r_a (\xi_1 - \mathcal{X})^{-1} a^{-1}$. However, since c is chosen uniformly at random and is independent from $Z, R_1, R_2, u, U, \mathsf{ct}'$, the equation on c only holds with negligible probability. Therefore $Z = \mathsf{ct}_1^{\mathcal{X}}$ and $R_2 = R_1^{\mathcal{X}}$ with overwhelming probability. $\mathcal{B}$ returns $M = \mathsf{ct}'_2/Z$.
- To check π_{sub}: parse $\pi_{\mathsf{sub}} = (\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2), \varrho_a, \varrho_k, \{e_j, \varrho_j\}_{j=1}^{v})$. Similarly, $\mathcal{B}$ looks throughout $\mathcal{H}$ to find the unique query $(\mathsf{ct}, Z, R_1, R_2, R_3, A_1, \cdots, A_v, B_1, \cdots, B_v)$ that satisfies all the equality tests in the verification. $\mathcal{B}$ returns $M = \mathsf{ct}_2/Z$.

$\mathcal{A}$ outputs a guess bit $\hat{b}$ and $\mathcal{B}$ outputs $b \overset{?}{=} \hat{b}$. In this simulation, when $\mathcal{Z} = \mathcal{X}\mathcal{Y}$, from $\mathcal{A}$'s point of view, $\mathcal{A}$ plays against Game 0 and therefore $\Pr[S_0] = \Pr[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{X}\mathcal{Y}}) = 1]$. When $\mathcal{Z}$ is a uniform random, ct_2 is uniformly distributed and $\mathcal{A}$ plays against Game 1. Hence $\Pr[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}}) = 1] = \Pr[S_1]$. By XDH assumption, we have that $|\Pr[S_1] - \Pr[S_0]| = |\Pr[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}}) = 1] - \Pr[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{X}\mathcal{Y}}) = 1]| \leq \mathsf{negl}(\lambda)$.
Game 2. Let Game 2 be the same as Game 1, except $(\mathsf{aid}^*, \pi^*_{\mathsf{rep}})$ in the challenge σ^* is simulated as described in Lemma 2. Let S_2 be the event that $b = \hat{b}$ in this game. We shall prove that $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\lambda)$ under the XDH assumption in the random oracle model. Assume an adversary $\mathcal{A}$ breaks the anonymity of the pRate scheme. We shall construct an adversary $\mathcal{B}$ which breaks the XDH assumption.
Let $(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}})$ be an XDH problem given to $\mathcal{B}$ where $\mathbb{G}_1$ is assumed to be associated with a bilinear map $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathcal{B}$'s goal is to decide whether $\mathcal{Z} = \mathcal{X}\mathcal{Y}$. $\mathcal{B}$ generates parameters for pRate scheme as follows: $\gamma, \xi \xleftarrow{\$} \mathbb{Z}_p^*$, $g_0, g_1, \cdots, g_{v+1}, g_{v+3}, g, u \xleftarrow{\$} \mathbb{G}_1$, $w \xleftarrow{\$} \mathbb{G}_2$. Compute $W = w^{\gamma}$ and $U = u^{\xi}$. Let $g_{v+2} = g'$. The master issuing key is $\mathsf{ik} = \gamma$ and the master opening key is $\mathsf{ok} = \xi$. The public parameters are $\mathsf{pp} = (g_0, g_1, \cdots, g_{v+3}, g, w, W, u, U)$. $\mathcal{B}$ gives pp and ik to $\mathcal{A}$. Suppose $\mathcal{A}$ queries to add at most q_A users. $\mathcal{B}$ chooses $i_0^*, i_1^* \xleftarrow{\$} \{1, \cdots, q_A\}$. $\mathcal{B}$ fixes i_0^* as the challenge query. Below we describe how $\mathcal{B}$ answers the queries on i_0^* and the other queries can be easily answered.

– On a $\mathsf{AddU}(i_0^*)$ query, $\mathcal{B}$ sets $K = {g'}^{\mathcal{X}}$ in Join and Issue protocol. Since $\mathcal{B}$ does not know $\mathcal{X}$, $\mathcal{B}$ sets $\mathsf{urep}[i_0^*] = (\mathsf{nil}, s, e, R, C)$.
– On a $\mathsf{RepAds}(i_0^*, m, P)$ query, $\mathcal{B}$ has to simulate $(\mathsf{aid}, \pi_{\mathsf{rep}})$ since it does not know $\mathcal{X}$. $\mathcal{B}$ chooses $a \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\mathsf{aid} = ({g'}^a, K^a)$. The proof π^*_{rep} is simulated as described in Lemma 2.
– On Token and SndTok queries that are related to user i_0^*, since $\mathcal{B}$ does not know $\mathcal{X}$, $\mathcal{B}$ sets $\mathsf{ct} = (u^a, {g'}^{\mathcal{X}} \cdot U^a)$ for user i_0^* and simulates the corresponding proofs π_{enc} and π_{tok} as described in Game 1.
– On the challenge query, $\mathcal{A}$ submits two indices i_0, i_1. Then $\mathcal{B}$ chooses a random bit $b \xleftarrow{\$} \{0, 1\}$. If $i_b \neq i_0^*$ or $i_{1-b} \neq i_1^*$ then $\mathcal{B}$ aborts and outputs Fail. Otherwise $\mathcal{B}$ creates a challenge $\sigma^* = (\mathsf{aid}^*, \pi^*_{\mathsf{rep}}, \mathsf{sn}', \mathsf{RT}', \mathsf{UT}', \delta)$ in the same way as in Game 1 except that $\mathsf{aid}^* = ({g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}})$ and π^*_{rep} is simulated as described in Lemma 2.

$\mathcal{A}$ outputs a guess bit $\hat{b}$ and $\mathcal{B}$ outputs $b \overset{?}{=} \hat{b}$. Since i_0^*, i_1^* are uniformly and independently drawn, the probability that Fail does not occur is $1/q_A^2$. In this simulation, when $\mathcal{Z} = \mathcal{X}\mathcal{Y}$, $\mathcal{A}$ plays against Game 1 and therefore $\Pr[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{X}\mathcal{Y}}) = 1] = \Pr[S_1]/q_A^2$. When $\mathcal{Z}$ is a uniform random, aid^* is uniformly distributed and

$\mathcal{A}$ plays against Game 2. Hence $\mathsf{Pr}[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}}) = 1] = \mathsf{Pr}[S_2]/q_A^2$. By XDH assumption, we have that $|\mathsf{Pr}[S_2] - \mathsf{Pr}[S_1]| = q_A^2 |\mathsf{Pr}[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}}) = 1] - \mathsf{Pr}[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{XY}}) = 1]| \leq \mathsf{negl}(\lambda)$. Moreover, $\mathcal{B}$'s answers to all of $\mathcal{A}$'s queries contain no info about b, hence we have $\mathsf{Pr}[S_2] = 1/2$.

Finally, from $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anony}}(1^\lambda) = |\mathsf{Pr}[S_0] - \frac{1}{2}|$, $|\mathsf{Pr}[S_1] - \mathsf{Pr}[S_0]| \leq \mathsf{negl}(\lambda)$, $|\mathsf{Pr}[S_2] - \mathsf{Pr}[S_1]| \leq \mathsf{negl}(\lambda)$ and $\mathsf{Pr}[S_2] = 1/2$, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{anony}}(1^\lambda) = |\mathsf{Pr}[S_0] - \frac{1}{2}| = |\mathsf{Pr}[S_0] - \mathsf{Pr}[S_1] + \mathsf{Pr}[S_1] - \mathsf{Pr}[S_2]| \leq |\mathsf{Pr}[S_0] - \mathsf{Pr}[S_1]| + |\mathsf{Pr}[S_1] - \mathsf{Pr}[S_2]| \leq \mathsf{negl}(\lambda)$.

Theorem 2. *The pRate scheme is rating-secret under the XDH assumption.*

Proof. The proof proceeds with a sequence of games.

Game 0. Let Game 0 be the original game $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{secrecy}}(1^\lambda)$ and let S_0 be the event that $\hat{b} = b$ in this game. Clearly we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{secrecy}}(1^\lambda) := |\mathsf{Pr}[S_0] - \frac{1}{2}|$.

Game 1. Let Game 1 be the same as Game 0 except $V, \mathsf{sn} \xleftarrow{\$} \mathbb{G}_1$ and π_{sub} is simulated in the challenge δ^*. Let S_1 be the event that $b = \hat{b}$ in this game. We shall prove that $|\mathsf{Pr}[S_1] - \mathsf{Pr}[S_0]| \leq \mathsf{negl}(\lambda)$ under the XDH assumption in the random oracle model.

Assume an adversary $\mathcal{A}$ breaks the anonymity of the pRate scheme. We shall contruct an adversary $\mathcal{B}$ which breaks the XDH assumption. Let $(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}})$ be an XDH problem given to $\mathcal{B}$ where $\mathbb{G}_1$ is assumed to be associated with a bilinear map $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathcal{B}$'s goal is to decide whether $\mathcal{Z} = \mathcal{XY}$. $\mathcal{B}$ generates parameters for pRate scheme as follows: $\gamma, \xi \xleftarrow{\$} \mathbb{Z}_p^*$, $g_0, g_1, \cdots, g_{v+3}, u \xleftarrow{\$} \mathbb{G}_1$ and $w \xleftarrow{\$} \mathbb{G}_2$. Compute $W = w^\gamma$ and $U = u^\xi$. Let $g = g'$ and $g_{v+3} = {g'}^{\mathcal{X}}$. The master issuing key is $\mathsf{ik} = \gamma$ and the master opening key is $\mathsf{ok} = \xi$. The public parameters are $\mathsf{pp} = (g_0, g_1, \cdots, g_{v+3}, g, w, W, u, U)$. $\mathcal{B}$ gives $\mathsf{pp}, \mathsf{ik}, \mathsf{ok}$ to $\mathcal{A}$. Suppose $\mathcal{A}$'s queries lead to at most q_A items in the list RL. $\mathcal{B}$ chooses $k^* \xleftarrow{\$} \{1, \cdots, q_A\}$ and fixes k^* as the challenge query. Since $\mathcal{B}$ has ik and ok, $\mathcal{B}$ can easily answer the queries AddU, RReg, CrptU, RepAds, Token, $\mathsf{RateGen}$, $\mathsf{RateUpd}$. Below we describe how $\mathcal{B}$ answers the query for generating $\mathsf{RL}[k^*]$ and the challenge δ^*.

- Assume $\mathsf{Token}(j_0, j_1)$ is the query that creates the k^*th item in RL. Let $\mathsf{AL}[j_0] = (i_0, \mathsf{aid}_0)$ and $\mathsf{AL}[j_1] = (i_1, \mathsf{aid}_1)$. W.l.o.g., assume i_0 will be put in $\mathsf{RL}[k^*]$. Since $\mathcal{B}$ does not know the value of $\mathcal{Y}$, $\mathcal{B}$ simulates the running of $(\mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_0], \mathsf{aid}_0, \mathsf{aid}_1), \mathsf{Token}(\mathsf{pp}, \mathsf{urep}[i_1], \mathsf{aid}_1, \mathsf{aid}_0))$ by setting $\mathsf{sn} = {g'}^{\mathcal{Y}}$ and $\mathsf{RT} = (\mathsf{sn}, \mathsf{nil}, a, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}})$ and $\mathsf{UT}' = \mathsf{nil}$. $\mathcal{B}$ sets $\mathsf{RL}(k^*) = (i_0, \mathsf{sn}, \mathsf{RT}, 0)$.
- On the challenge query, $\mathcal{A}$ submits (x_0, x_1, k). If $k \neq k^*$ then $\mathcal{B}$ aborts and outputs Fail. To create the challenge δ^*, $\mathcal{B}$ chooses $b \xleftarrow{\$} \{0, 1\}$ and simulates $\mathsf{RateGen}$ by setting $V = g_{x_b} \cdot {g'}^{\mathcal{Z}}$ and $\delta^* = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$, where $\mathsf{RT} = (\mathsf{sn}, \mathsf{nil}, a, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}})$ and π_{sub} is a simulated proof.

$\mathcal{A}$ outputs a guess bit $\hat{b}$ and $\mathcal{B}$ outputs $b \overset{?}{=} \hat{b}$. Since k^* is chosen uniformly at random, the probability that Fail does not occur is $1/q_A$. In this simulation, when $\mathcal{Z} = \mathcal{XY}$, from $\mathcal{A}$'s point of view, $\mathcal{A}$ plays against Game 0. Therefore $\mathsf{Pr}[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{XY}}) = 1] = \mathsf{Pr}[S_0]/q_A$. When $\mathcal{Z}$ is chosen uniformly at random, sn, V are randomly distributed and thus V contains no information of b. Therefore $\mathcal{A}$ plays against Game 1 and $\mathsf{Pr}[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}}) = 1] = \mathsf{Pr}[S_1]/q_A$. Moreover, $\mathcal{B}$'s answers to all of $\mathcal{A}$'s queries contain no info about b, hence we have $\mathsf{Pr}[S_1] = 1/2$. By XDH assumption, we have $|\mathsf{Pr}[S_0] - \mathsf{Pr}[S_1]| = |\mathsf{Pr}[S_0] - 1/2| = q_A |\mathsf{Pr}[\mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{XY}})] - \mathcal{B}(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}}, {g'}^{\mathcal{Y}}, {g'}^{\mathcal{Z}}) = 1| \leq \mathsf{negl}(\lambda)$.

Theorem 3. *The pRate scheme is advertisement-unforgeable under the q-SDH assumption.*

Proof. Assume an adversary $\mathcal{A}$ breaks the traceability of pRate scheme. We shall construct an adversary $\mathcal{B}$ which breaks the q-SDH assumption, where q is the upper bound of the number of reputation credentials generated by $\mathcal{A}$ through the oracle queries. Let $(g', {g'}^{\gamma}, \cdots, {g'}^{\gamma^q}, g'_2, {g'_2}^{\gamma})$ be a random instance of the q-SDH problem. The goal of $\mathcal{B}$ is to find a pair $({g'}^{\frac{1}{\gamma+e}}, e)$ for some $e \in \mathbb{Z}_p^*$. $\mathcal{B}$ generates the parameters of pRate scheme as follows: $\xi, \zeta_1, \cdots, \zeta_{v+3}, e_1, \cdots, e_q, z_1, \cdots, z_q, \theta, \theta_1, \cdots, \theta_q \xleftarrow{\$} \mathbb{Z}_p^*$, $g, u \xleftarrow{\$} \mathbb{G}_1$, $w \xleftarrow{\$} \mathbb{G}_2$. Define polynomials

$$F(X) = \prod_{i=1}^{q} (X + e_i) = \sum_{j=0}^{q} a_j X^j$$

$$F_i(X) = F(X)/(X+e_i) = \prod_{j=1, j\neq i}^{q} (X+e_i) = \sum_{j=0}^{q-1} a_{i,j} X^j$$

$$F_{i,j}(X) = \frac{F(X)}{(X+e_i)(X+e_j)} = \prod_{\iota=1, \iota\neq i,j}^{q} (X+e_j) = \sum_{\iota=0}^{q-2} a_{i,j,\iota} X^\iota \text{ for } i \neq j$$

Set

$$g_0 = (\prod_{i=0}^{q} {g'_1}^{a_i \gamma^i})^\theta \prod_{i=1}^{q} (\prod_{j=0, j\neq i}^{q-1} {g'_1}^{a_{i,j}\gamma^j})^{\theta_i z_i} = {g'_1}^{\theta F(\gamma) + \sum_{i=1}^{q} \theta_i z_i F_i(\gamma)}$$

$$H = \prod_{i=1}^{q} (\prod_{j=0, j\neq i}^{q-1} {g'_1}^{a_{i,j}\gamma^j})^{\theta_i} = {g'_1}^{\sum_{i=1}^{q} \theta_i F_i(\gamma)}$$

$$g_1 = H^{-\zeta_1}, g_2 = H^{-\zeta_2}, \cdots, g_{v+3} = H^{-\zeta_{v+3}}, w = g'_2, W = {g'_2}^{\gamma} \text{ and } U = u^\xi$$

The master issuing key is $\mathsf{ik} = \gamma$ which $\mathcal{B}$ does not have, and the master opening key $\mathsf{ok} = \xi$. The public parameters are $\mathsf{pp} = (g_0, g_1, \cdots, g_{v+3}, g, w, W, u, U)$. We shall describe how to answer AddU, $\mathsf{RateUpd}$ queries and the other queries can be easily answered.

– For an $\mathsf{AddU}(i)$ query, $\mathcal{B}$ chooses unselected $e_{i'} \in \{e_1, \cdots, e_q\}$ and $z_{i'} \in \{z_1, \cdots, z_q\}$, $k_{i'} \xleftarrow{\$} \mathbb{Z}_p^*$, a timestamp t and initial values $n_1, \cdots, n_v$. Compute

$$s = (z_{i'} - (n_1 \cdot \zeta_1 + \cdots + n_v \cdot \zeta_v + t \cdot \zeta_{v+1} + k_{i'} \cdot \zeta_{v+2}))/\zeta_{v+3}$$

$$R = g_0 H^{-z_{i'}} = g_0 g_1^{n_1} \cdots g_v^{n_v} g_{v+1}^{t} g_{v+2}^{k_{i'}} g_{v+3}^{s} = {g'_1}^{\theta F(\gamma) + \sum_{j\in[1,q]}^{j\neq i'} \theta_j (z_j - z_{i'}) F_j(\gamma)}$$

$$C_{i'} = (\prod_{j=0}^{q-1} {g'_1}^{a_{i',j}\gamma^j})^\theta \prod_{j\in[1,q]}^{j\neq i'} (\prod_{\iota=0}^{q-2} {g'_1}^{a_{j,i',\iota}\gamma^\iota})^{\theta_j(z_j - z_{i'})} = {g'_1}^{\theta F_{i'}(\gamma) + \sum_{j\in[1,q]}^{j\neq i'} \theta_j (z_j - z_{i'}) F_{j,i'}(\gamma)} = R^{\frac{1}{\gamma + e_{i'}}}$$

Set $\mathsf{reg}[i] = (K, e_{i'}, t, R, C_{i'})$, $\mathsf{urep}[i] = (k_{i'}, s, e_{i'}, R, C_{i'})$, $\mathsf{scr}[i] = (n_1, \cdots, n_v, t)$ where $K = g_{v+2}^{k_{i'}}$.

– For $\mathsf{RateUpd}(\mathsf{sn}, \delta, M_{\mathsf{in}})$ query with $\delta = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$, $\mathcal{B}$ checks the validity of $\pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}}$ and extracts the identities of rater and ratee from ct and ct'. If successful, $\mathcal{B}$ goes through UL and find ι s.t. $\mathsf{UL}[\iota] = (\mathsf{sn}, \mathsf{UT}')$ and $g^{\mathsf{UT}'} = \mathsf{sn}$. If no such ι, $\mathcal{B}$ checks if $g^{M_{\mathsf{in}}} \overset{?}{=} \mathsf{sn}$. If not, $\mathcal{B}$ aborts. Otherwise set $\mathsf{UT}' = M_{\mathsf{in}}$. $\mathcal{B}$ uses UT' to extract the rating in V by computing $g_x = V/g_{v+3}^{\mathsf{UT}'}$. Suppose the rater is i and the ratee is j. Let $\mathsf{reg}[j] = (K, e, t, R, C)$, $\mathsf{urep}[j] = (k, s, e, R, C)$ and $\mathsf{scr}[j] = (n_1, \cdots, n_v, t)$. $\mathcal{B}$ chooses unselected $e_{i'} \in \{e_1, \cdots, e_q\}$, $z_{i'} \in \{z_1, \cdots, z_q\}$ and a new timestamp t'. Compute

$$s' = (z_{i'} - (n_1 \cdot \zeta_1 + \cdots + (n_x + 1) \cdot \zeta_x + \cdots + n_v \cdot \zeta_v + t' \cdot \zeta_{v+1} + k \cdot \zeta_{v+2}))/\zeta_{v+3}$$

$$\begin{aligned} R' &= R \cdot g_{v+1}^{t'-t} \cdot V \cdot g_{v+3}^{s'-s-\mathsf{UT}'} = g_0 g_1^{n_1} \cdots g_x^{n_x+1} \cdots g_v^{n_v} g_{v+1}^{t'} g_{v+2}^{k} g_{v+3}^{s'} \\ &= g_0 H^{-z_{i'}} = {g'_1}^{\theta F(\gamma) + \sum_{j=1, j\neq i'}^{q} \theta_j (z_j - z_{i'}) F_j(\gamma)} \\ C_{i'} &= (\prod_{\iota=0}^{q-1} {g'_1}^{a_{i',\iota}\gamma^\iota})^\theta \prod_{j=1, j\neq i'}^{q} (\prod_{\iota=0}^{q-2} {g'_1}^{a_{j,i',\iota}\gamma^\iota})^{\theta_j(z_j - z_{i'})} \\ &= {g'_1}^{\theta F_{i'}(\gamma) + \sum_{j=1, j\neq i'}^{q} \theta_j (z_j - z_{i'}) F_{j,i'}(\gamma)} = {R'}^{\frac{1}{\gamma + e_{i'}}} \end{aligned}$$

Update user j's reputation credentials as $\mathsf{scr}[j] = (n_1, \cdots, n_x + 1, \cdots, n_v, t')$, $\mathsf{reg}[j] = (K, e', t', R', C_{i'})$ and $\mathsf{urep}[j] = (k, s', e', R', C_{i'})$.

$\mathcal{A}$ outputs $(\mathsf{aid}, \pi_{\mathsf{rep}}, m, P, n, x)$. Parse $\pi_{\mathsf{rep}} = (E, A, S, \eta, c, z_e, z_t, z_k, z_\alpha, z_\rho, z_s, \{z_{n_j}\}_{j=1}^{v}, z_{f_1}, z_{f_2}, \mathbf{l}, \mathbf{r})$. As described in Lemma 3, $\mathcal{B}$ rewinds $\mathcal{A}$ to the point just before the challenge c and obtain $\mathcal{B}$ extracts the tuple $(C^*, \Delta n_1, \cdots, \Delta n_v, \Delta t, \Delta k, \Delta s, \Delta\alpha, \Delta e)$ such that $C^* = E g_{v+3}^{-\Delta\alpha} = \left(g_0 g_1^{\Delta n_1} \cdots g_v^{\Delta n_v} g_{v+1}^{\Delta t} g_{v+2}^{\Delta k} g_{v+3}^{\Delta s}\right)^{\frac{1}{\gamma + \Delta e}}$. Let $\Delta z = \Delta n_1 \cdot \zeta_1 + \cdots + \Delta n_v \cdot \zeta_v + \Delta t \cdot \zeta_{v+1} + \Delta k \cdot \zeta_{v+2} + \Delta s \cdot \zeta_{v+3}$. Then we have $C^* = (g_0 H^{-\Delta z})^{\frac{1}{\gamma + \Delta e}} = {g'_1}^{\frac{\theta F(\gamma) + \sum_{j=1}^{q} \theta_j (z_j - \Delta z) F_k(\gamma)}{\gamma + \Delta e}}$.

Assume the indices $\{i_1, i_2, \cdots, i_{q'}\}$ are chosen for the queries to the oracle AddU with $q' \leq q$. In the experiment of defining advertisement-unforgeability, since there is no $i \in \{i_1, i_2, \cdots, i_{q'}\}$ s.t. the output of $\mathsf{RateAcc}$ equals to $(*, i, *)$ after running $\mathsf{Token}, \mathsf{RateGen}$, this means $\Delta k \notin \{k_{i_1}, \cdots, k_{i_q}\}$. Note that the $\mathsf{RateUpd}$ queries do not modify these k_{i_j}. Therefore there is no such $j \in \{1, \cdots, q\}$ s.t. $\Delta e = e_j$ and $C^* = C_j$ and $(\Delta n_1, \cdots, \Delta n_v, \Delta t, \Delta k, \Delta s) = (n_{1,j}, \cdots, n_{v,j}, t_j, k_j, s_j)$.

- If $\Delta e \notin \{e_1, \cdots, e_q\}$, $\mathcal{B}$ can obtain a new q-SDH pair as follows. Let $Q(X) = \theta F(X) + \sum_{j=1}^{q} \theta_j (z_j - \Delta z) F_k(X)$. We can compute $Q'(X)$ s.t. $Q(X) = (X + \Delta e)Q'(X) + Q(-\Delta e)$ with $Q(-\Delta e) \in \mathbb{Z}_p^*$. Since $\Delta e \notin \{e_1, \cdots, e_q\}$ and $\theta, \theta_1, \cdots, \theta_q$ are chosen uniformly at random, we know that $Q(-\Delta e) = 0$ with only negligible probability. Let $Q'(X) = \sum_{i=0}^{q-1} \beta_i X^i$. Compute $S = (C^* {g'_1}^{-\sum_{i=0}^{q-1} \beta_i \gamma^i})^{\frac{1}{Q(-\Delta e)}} = {g'_1}^{\frac{1}{\gamma + \Delta e}}$. $\mathcal{B}$ outputs $(S, \Delta e)$ as a solution to the q-SDH instance.
- If $\Delta e = e_i$ but $C^* \neq C_i$ for some $i \in \{1, \cdots, q\}$. This means $\Delta z \neq z_i$. We can obtain a new q-SDH pair as follows. We have $C^* = {g'_1}^{\frac{\theta F(\gamma) + \sum_{j=1}^{q} \theta_j (z_j - \Delta z) F_j(\gamma)}{\gamma + \Delta e}} = {g'_1}^{\theta F_i(\gamma) + \left(\sum_{j=1, j\neq i}^{q} \theta_j (z_j - \Delta z) F_{j,i}(\gamma)\right) + \frac{\theta_i (z_i - \Delta z) F_i(\gamma)}{\gamma + e_i}}$. We can compute $Q(X)$ s.t. $F_i(X) = (X + e_i)Q(X) + F_i(-e_i)$. Let $a^* = \theta_i (z_i - \Delta z) F_i(-e_i)$. Since $\Delta z \neq z_i$ and θ_i is a uniform random, we know that $a^* \neq 0$. Let $Q(X) = \sum_{j=0}^{q-1} \beta_j X^j$. Compute

$$S = (C^* {g'_1}^{-\theta \sum_{j=0}^{q-1} a_{i,j} \gamma^j} {g'_1}^{-\sum_{j=1, j\neq i}^{q} \theta_j (z_j - \Delta z) \sum_{\iota=0}^{q-2} a_{j,i,\iota} \gamma^\iota} {g'_1}^{-\theta_i (z_i - \Delta z) \sum_{j=0}^{q-1} \beta_j \gamma^j})^{\frac{1}{a^*}} = {g'_1}^{\frac{1}{\gamma + e_i}}$$

$\mathcal{B}$ outputs (S, e_i) as a solution to the q-SDH instance.
- If $\Delta e = e_i$ and $C^* = C_i$ for some $i \in \{1, \cdots, q\}$ but $(\Delta n_1, \cdots, \Delta n_v, \Delta t, \Delta k, \Delta s) \neq (n_1, \cdots, n_v, t, k, s)$ where $C_i = \left(g_0 g_1^{n_1} \cdots g_v^{n_v} g_{v+1}^t g_{v+2}^k g_{v+3}^s\right)^{\frac{1}{\gamma + e_i}}$. Then we can deduce obtain a non-trivial relation $\zeta_1(n_1 - \Delta n_1) + \cdots + \zeta_v(n_v - \Delta n_v) + \zeta_{v+1}(t - \Delta t) + \zeta_{v+2}(k - \Delta k) + \zeta_{v+3}(s - \Delta s) = 0$ on $\zeta_1, \cdots, \zeta_{v+3}$. Since $\zeta_1, \cdots, \zeta_{v+3}$ are chosen independently and randomly, this breaks DL-assumption.

Theorem 4. *The pRate scheme is ratee-unforgeable under the DL assumption.*

Proof. Assume an adversary $\mathcal{A}$ breaks the unforgeability on rating-token of pRate scheme. We shall construct an adversary $\mathcal{B}$ which breaks the DL assumption.

Let $(\mathbb{G}_1, p, g', {g'}^{\mathcal{X}})$ be a DL problem given to $\mathcal{B}$. The goal of $\mathcal{B}$ is to find out the value of $\mathcal{X}$. $\mathbb{G}_1$ is assumed to be associated with a bilinear map $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathcal{B}$ generates parameters for pRate scheme as follows: $\gamma, \xi \xleftarrow{\$} \mathbb{Z}_p^*$, $g_0, g_1, \cdots, g_{v+1}, g_{v+3}, g, u \xleftarrow{\$} \mathbb{G}_1$, $w \xleftarrow{\$} \mathbb{G}_2$, $W = w^\gamma$ and $U = u^\xi$. Let $g_{v+2} = g'$. The master issuing key is $\mathsf{ik} = \gamma$ and the master opening key is $\mathsf{ok} = \xi$. The public parameters are $\mathsf{pp} = (g_0, g_1, \cdots, g_{v+3}, g, w, W, u, U)$. $\mathcal{B}$ gives $\mathsf{pp}, \mathsf{ik}, \mathsf{ok}$ to $\mathcal{A}$. Suppose $\mathcal{A}$ queries to add at most q_A users. $\mathcal{B}$ chooses $j^* \xleftarrow{\$} \{1, \cdots, q_A\}$. Below we describe how $\mathcal{B}$ answers the queries on j^* and the other queries can be easily answered.

– On a $\mathsf{AddU}(j^*)$ query, $\mathcal{B}$ sets $K = {g'}^{\mathcal{X}}$ in Join and Issue protocol. Since $\mathcal{B}$ does not know $\mathcal{X}$, $\mathcal{B}$ sets $\mathsf{urep}[j^*] = (\mathsf{nil}, s, e, R, C)$.

– On a $\mathsf{RepAds}(j^*, m, P)$ query, $\mathcal{B}$ has to simulate $(\mathsf{aid}, \pi^*_{\mathsf{rep}})$ since it does not know $\mathcal{X}$. $\mathcal{B}$ chooses $a \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\mathsf{aid} = ({g'}^a, K^a)$. The proof π^*_{rep} is simulated as described in Lemma 2.

Since $\mathcal{A}$ is not allowed to query Token and SndTok on user j^*, there is no item in RL that is related to j^*. Finally $\mathcal{A}$ outputs δ. Parse $\delta = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$. $\mathcal{B}$ runs $\theta \xleftarrow{\$} \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$ which checks the validity of $\pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}}$ and extracts the identities of rater and ratee and accumulates ratings. Parse $\theta = (i, j, \mathsf{aux})$. If $j \neq j^*$ then $\mathcal{B}$ outputs Fail. Since j^* is uniformly and independently drawn, the probability that Fail does not occur is $1/q_A$. Parse $\pi'_{\mathsf{tok}} = (\mathsf{ct}' = (\mathsf{ct}'_1, \mathsf{ct}'_2), c, \varrho_a, \varrho_k)$. From $\theta = (i, j, \mathsf{aux})$, we know that $\mathsf{ct}'_2/{\mathsf{ct}'_1}^{\xi} = {g'}^{\mathcal{X}}$. Hence π'_{tok} is a valid proof and $\mathcal{B}$ never simulates such a proof. $\mathcal{B}$ uses Forking Lemma on π'_{tok} and rewinds $\mathcal{A}$ to the point just before the challenge c. To challenge c', suppose $\mathcal{A}$ generates ϱ'_a, ϱ'_k. Let $\Delta a = (\varrho_a - \varrho'_a)/(c - c')$ and $\Delta k = (\varrho_k - \varrho'_k)/(c - c')$. Using the equations in the verification algorithm of π'_{tok}, we obtain that $u^{\Delta a} = \mathsf{ct}'_1$ and $g_{v+2}^{\Delta k} U^{\Delta a} = \mathsf{ct}'_2$. Therefore, $\mathsf{ct}'_2/{\mathsf{ct}'_1}^{\xi} = g_{v+2}^{\Delta k} = g_{v+2}^{\mathcal{X}}$ which means $\mathcal{B}$ finds a solution $\mathcal{X} = \Delta k$ to the DL problem.

Theorem 5. *The pRate scheme is rater-unforgeable under the DL assumption.*

Proof. Assume an adversary $\mathcal{A}$ breaks the unforgeability on rating-token of pRate scheme. We shall construct an adversary $\mathcal{B}$ which breaks the DL assumption.

Let $(\mathbb{G}_1, p, g', g'^{\mathcal{X}})$ be a DL problem given to $\mathcal{B}$. The goal of $\mathcal{B}$ is to find out the value of $\mathcal{X}$. $\mathbb{G}_1$ is assumed to be associated with a bilinear map $\hat{\mathsf{e}} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. $\mathcal{B}$ generates parameters for pRate scheme as follows. $\gamma, \xi \xleftarrow{\$} \mathbb{Z}_p^*$, $g_0, g_1, \cdots, g_{v+1}, g_{v+3}, g, u \xleftarrow{\$} \mathbb{G}_1$, $w \xleftarrow{\$} \mathbb{G}_2$. Compute $W = w^{\gamma}$ and $U = u^{\xi}$. Let $g_{v+2} = g'$. The master issuing key is $\mathsf{ik} = \gamma$ and the master opening key is $\mathsf{ok} = \xi$. The public parameters are $\mathsf{pp} = (g_0, g_1, \cdots, g_{v+3}, g, w, W, u, U)$. $\mathcal{B}$ gives $\mathsf{pp}, \mathsf{ik}, \mathsf{ok}$ to $\mathcal{A}$. Suppose $\mathcal{A}$ queries to add at most q_A users. $\mathcal{B}$ chooses $i^* \xleftarrow{\$} \{1, \cdots, q_A\}$. Below we describe how $\mathcal{B}$ answers the queries on i^* and the other queries can be easily answered.

- On a $\mathsf{AddU}(i^*)$ query, $\mathcal{B}$ sets $K = g'^{\mathcal{X}}$ in Join and Issue protocol. Since $\mathcal{B}$ does not know $\mathcal{X}$, $\mathcal{B}$ sets $\mathsf{urep}[i^*] = (\mathsf{nil}, s, e, R, C)$.
- On a $\mathsf{RepAds}(i^*, m, P)$ query, $\mathcal{B}$ has to simulate $(\mathsf{aid}, \pi_{\mathsf{rep}})$ since it does not know $\mathcal{X}$. $\mathcal{B}$ chooses $a \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\mathsf{aid} = (g'^{a}, K^{a})$. The proof π^*_{rep} is simulated as described in Lemma 2.
- On Token and SndTok queries that are related to user i^*, $\mathcal{B}$ sets $\mathsf{ct} = (u^{a}, g'^{\mathcal{X}} \cdot U^{a})$ for user i^* and simulates the proofs π_{enc} and π_{tok} since $\mathcal{B}$ does not know $\mathcal{X}$.

$\mathcal{A}$ outputs δ. Parse $\delta = (\mathsf{sn}, V, \mathsf{ct}, \mathsf{ct}', \mathsf{sn}', \pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}})$. $\mathcal{B}$ runs $\theta \xleftarrow{\$} \mathsf{RateAcc}(\mathsf{pp}, \mathsf{ok}, \mathsf{reg}, \delta)$ which checks the validity of $\pi'_{\mathsf{tok}}, \pi_{\mathsf{sub}}$ and extracts the identities of rater and ratee and accumulates ratings. Parse $\theta = (i, j, \mathsf{aux})$. If $i \neq i^*$ then $\mathcal{B}$ outputs Fail. Since i^* is uniformly and independently drawn, the probability that Fail does not occur is $1/q_A$. Parse $\pi_{\mathsf{sub}} = (\mathsf{ct}, \varrho_a, \varrho_k, \{e_j, \varrho_j\}_{j=1}^{v})$. From $\theta = (i, j, \mathsf{aux})$, we know that $\mathsf{ct}_2/\mathsf{ct}_1{}^{\xi} = g'^{\mathcal{X}}$. Note that π_{sub} is a valid proof and $\mathcal{B}$ never simulates such a proof. $\mathcal{B}$ uses Forking Lemma on π_{sub} and rewinds $\mathcal{A}$ to the point just before the challenge c. To challenge c', suppose $\mathcal{A}$ generates $\varrho'_a, \varrho'_k, e'_x, \varrho'_x$. Let $\Delta a = (\varrho_a - \varrho'_a)/(c - c')$, $\Delta k = (\varrho_k - \varrho'_k)/(c - c')$ and $\Delta r = (\varrho_x - \varrho'_x)/(e_x - e'_x)$. Using the equations in the verification algorithm of π_{sub}, we obtain that $u^{\Delta a} = \mathsf{ct}_1$ and $g_{v+2}^{\Delta k} U^{\Delta a} = \mathsf{ct}_2$. Therefore, $\mathsf{ct}_2/\mathsf{ct}_1^{\xi} = g_{v+2}^{\Delta k} = g_{v+2}^{\mathcal{X}}$ which means $\mathcal{B}$ finds a solution $\mathcal{X} = \Delta k$ to the DL problem.