DE GRUYTER OPEN Proceedings on Privacy Enhancing Technologies ..; .. (..):1–20

Sazzadur Rahaman*, Long Cheng, Danfeng (Daphne) Yao, He Li, and Jung-Min (Jerry) Park

Provably Secure Anonymous-yet-Accountable Crowdsensing with Scalable Sublinear Revocation

Abstract: Group signature schemes enable anonymous-yet-accountable communications. Such a capability is extremely useful for applications, such as smartphone-based crowdsensing and citizen science. However, the performance of modern group signature schemes is still inadequate to manage large dynamic groups. In this paper, we design the first provably secure verifier-local revocation (VLR) - based group signature scheme that supports sublinear revocation, named *Sublinear Revocation with Backward unlinkability and Exculpability (*SRBE*)*. To achieve this performance gain, SRBE introduces *time bound pseudonyms* for the signer. By introducing low-cost short-lived pseudonyms with sublinear revocation checking, SRBE drastically improves the efficiency of the group-signature primitive. The backward-unlinkable anonymity of SRBE guarantees that even after the revocation of a signer, her previously generated signatures remain unlinkable across epochs. This behavior favors the dynamic nature of real-world crowdsensing settings. We prove its security and discuss parameters that influence its scalability. Using SRBE, we also implement a prototype named GROUPSENSE for anonymous-yet-accountable crowdsensing, where our experimental findings confirm GROUPSENSE's scalability. We point out the open problems remaining in this space.

Keywords: Group Signature, Verifier Local Revocation, Privacy, Participatory Sensing, Crowdsensing.

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

***Corresponding Author: Sazzadur Rahaman:** Department of Computer Science, Virginia Tech, E-mail: sazzad14@cs.vt.edu
Long Cheng: Department of Computer Science, Virginia Tech, E-mail: chengl@cs.vt.edu
Danfeng (Daphne) Yao: Department of Computer Science, Virginia Tech, E-mail: danfeng@cs.vt.edu
He Li: Department of Electrical and Computer Engineering, Virginia Tech, E-mail: heli@ece.vt.edu
Jung-Min (Jerry) Park: Department of Electrical and Computer Engineering, Virginia Tech, E-mail: jungmin@ece.vt.edu

1 Introduction

The new urban-scale crowdsensing vision promises intriguing applications, such as health monitoring [1], environment monitoring [2], traffic prediction [3], etc. However, an open crowdsensing platform where anyone can submit data is undesirable. It is exposed to malicious and erroneous participation, which may threaten data integrity and reliability [4]. Accountability of participants for their data reports is a key requirement for crowdsensing platforms [5]. To serve this requirement, it is desirable that only the participants with proper authorization should be able contribute in a crowdsensing platform. We refer to this controlled crowdsensing scenario as *groupsensing*.

While accountability protects the data collector, the vast number of crowdsensing participants sharing sensitive information such as location, daily routine, health status, need to be protected against privacy threats [6, 7]. The threats may include the semi-honest data-collection service provider, who attempts to track and de-anonymize participants, as well as data breaches on the data-collection servers [8–10]. Therefore, the *sensing-time anonymity* is also an essential requirement, especially for the participants those are involved in long-term sensing applications.

Anonymous-yet-accountable crowdsensing demands "privacy preserving authentication". *Privacy preserving authentication* is a cryptographic protocol to authenticate users without knowing their identity [11]. These protocols can broadly be categorized into two groups: (1) *pseudonym-based systems* [12, 13] and (2) *group signature-based systems* [14–16]. Both of them rely on a trusted *group manager* to coordinate between the *signer* (*i.e.*, a crowdsensing participant) and the *verifier* (semi-honest data-collection server).

In pseudonym-based schemes (*e.g.*, [17]), the group manager needs to issue a list of pseudonyms and public-key certificates to certify the public keys of participants (for the accountability purpose). Participants generate signatures using pseudonyms and refresh pseudonyms periodically to preserve anonymity. However, all the signatures under the same pseudonym are linkable. Thus, frequent public-key certification and distribution are necessary to enable short-lived pseudonyms, which appears to be expensive [18].

In comparison to pseudonyms, group signature schemes (*e.g.*, [14–16, 19, 20]) do not require frequent public-key certifications for participants. One public key serves for all signatures of all participants. However, the trade-offs among security properties, computational and communication overheads [11, 21–23] for various applications has been the prime focus in the state-of-the-art group signature literature. In general, the revocation checking operation is known to be the most expensive operation for modern group signature schemes [24], which is necessary to enforce the blacklisting of misbehaving/deactivated users. As the revocation lists for these schemes are maintained locally at the verifiers' end, these schemes are known as verifier-local revocation (VLR). Typically, revocation lists contain revocation tokens, where a revocation token uniquely represents a revoked user. The computational complexity of deterministic revocation checking for VLR-based group signature schemes is typically $O(R)$, where R is the size of the revocation list.

Therefore, the attractiveness of group signatures (*e.g.*, [16, 22, 25, 26]) in crowdsensing is substantially dampened by the expensive revocation checking operations. For example, SPPEAR [27], a comprehensive crowdsensing system, avoids using group signatures for sensory data submission. SPPEAR only uses group signatures for setting up pseudonyms, but resorts to the public-key certification approach for data submissions. AnonySense [28] is another privacy-preserving crowdsensing framework that uses group signatures for data submissions. However, AnonySense does not support membership revocation. Thus, its accountability guarantee is low. Also, There exists several other proposals to preserve anonymity without accountability support [29–31].

In this paper, we present a new VLR-based group signature scheme named *Sublinear Revocation with Backward unlinkability and Exculpability (*SRBE*)*. SRBE's security is guaranteed under the random oracle model [16]. The main feature of SRBE is that the computational complexity of revocation check is $O(\log_2 R)$, where R is the size of the revocation list, which is explained next. Exculpability refers to that a group manager cannot forge a signature of any honest signer (*i.e.*, the private key of the signer is not compromised) that the signer cannot dispute.

In VLR-based group signature schemes, signatures carry zero-knowledge proofs of signers' revocation tokens [15, 16, 22], so that the revocation tokens are not available in the signature for direct comparison. To overcome this issue, SRBE uses *time bound pseudonyms* as revocation tokens. This approach enables verifiers to organize revocation tokens in standard data structures (i.e., binary search trees) for fast revocation check. The main technical challenge to use these time bound short-lived pseudonyms is, to embed them in signatures with minimal overheads as well as preserving the security properties.

SRBE's *anonymity* is defined in terms of *Backward Unlinkable anonymity (BU-anonymity)*, which means that even after the revocation of a signer, signatures produced by the signer before revocation remain anonymous across epochs. Our work supports unlinkability across different epochs [11]. Signatures generated by a signer in different time periods cannot be linked, as they are signed with unlinkable pseudonyms. Our pseudonyms are unique, as they support *i)* sublinear revocation and revocation checking, and *ii)* constant revocation token size per signer.

The requirement of constant revocation token size per signer is important. To revoke a signer, one needs to revoke all the pseudonyms of the signer. It would be inefficient if the size of the revocation token increases with the total number of pseudonyms per signer. We aim to keep the size of revocation tokens *constant*. Our pseudonyms are generated using a combination of forward and reverse cryptographic hash chains. In Section 3.1, we explain why some straightforward pseudonym attempts do not work.

In our work, signatures signed under the same pseudonym (within the same time period) can be linked by the verifier (i.e., the data-collection server), i.e., linkable. The similar limitation exists in other group signature schemes [11, 32]. Supporting the unlinkability within an epoch remains an open problem.

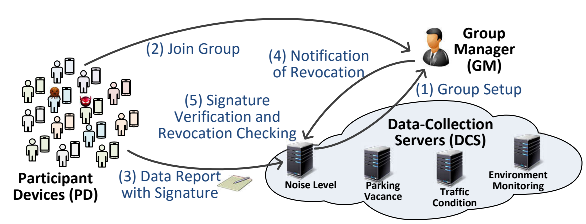


Fig. 1. System Overview of GroupSense

Using SRBE group signatures, we develop a groupsensing prototype named GroupSense. As illustrated in Figure 1, in GroupSense, a participant submits anonymously signed data reports to the data-collection server – the signature does not reveal her identity but proves her membership. The group manager can locate a participant for revocation purpose. The prototype includes client-side Android apps and server-side programs, where the performance of the prototype is also extensively evaluated.

Our technical contributions are summarized as follows.

– We present a new bilinear-map based group signature scheme (referred to as SRBE) with sub-linear revocation check. SRBE also supports backward unlinkability and exculpability. We prove the security in the random oracle model (ROM) using standard security assumptions.

– We implement SRBE group signature scheme, along with four other leading group signature schemes. We theoretically and experimentally compare their computation and communication costs and the scalability of revocation check algorithm. We also use SRBE to implement GROUPSENSE, an anonymous-yet-accountable crowdsensing prototype.
– Our experimental results indicate the scalability of GROUPSENSE with SRBE for large crowds. The major experimental findings are as follows. The revocation check procedure of SRBE gives around 3-order of magnitude performance gain over state-of-the-art. After precomputing some expensive operations, the average signing cost of SRBE in *Nexus 10* is around 1.69 second. GROUPSENSE takes around 150ms on average to verify a signature with revocation checking against $70,000$ of revoked users.

Multi-disciplinary crowdsensing and citizen-science projects [33] require secure and privacy-preserving cyberinfrastructures. Secure crowdsensing encourages participation, which in turn boosts the quality of data and discovery. We envision that the efficiency and scalability of SRBE may help increase the real-world adoption of group signatures by developers, scientists and engineers in crowdsensing and other applications requiring privacy.

2 Related Work

2.1 Revocation in Group Signatures

Group Signature (GS) Schemes allow signers (managed by some authority) to anonymously produce signatures on behalf of a group. After the introduction [14], different variants of GS schemes were proposed (collision-resistant GS scheme [19], sign-and-encrypt-and-prove based GS [20], traceable signatures [15, 34, 35], GS with verifier local revocation [16], GS from blind signatures [36], traceable signatures in standard model [37], group signatures in standard model [38], GS with controllable linkability [39, 40], GS with distributed traceability [41], GS with dynamic accumulators [42]). Kiayias *et al.* (*traceable signatures* [15]) first introduced internal tracing algorithms (trapdoors) to efficiently trace and revoke the anonymity of misbehaving members. They also formalized the properties of "Traceability" and "Exculpability" to extend existing security guarantees for better accountability.

Existing GS literature discussing different types of membership revocation procedure can be classified as followed.

Revocation in dynamic accumulators. Dynamic accumulator based schemes [42–45] provide constant time revocation check. However, the major disadvantage of these schemes is that, the signer needs to maintain a whitelist of unrevoked users to create a valid signature. This requires updating each of the signer's secret key on each revocation, which is impractical for large dynamic groups. To address this, Nakanishi *et al.* [23] proposed a revocable scheme with constant signing and verifying complexity, where no updates of secret key are required. However, this scheme still requires signers to fetch data of $O(R)$ complexity, where R is the total number of revoked users and the size of the public key is $O(\sqrt{N})$, where N is the total number of users.

Revocation in linking-based schemes. Recently, Slamanig *et al.* proposed a *Sign-and-Encrypt-and-Prove* based GS scheme with efficient revocation check [21] (similar as SRBE). The scheme adopts a centralized online OCSP service for revocation checking. In most of the *Sign-and-Encrypt-and-Prove* based GS schemes [21, 39, 40], RAs need to deanonymize a signature to perform revocation. However, as every signature verification needs a consultation with OCSP server for revocation checking, the communication overhead between verifier and the OCSP server becomes onerous. Most importantly, it is undesirable to deanonymize the signatures from benign users [27, 46], which might encourage massive surveillance [47]. Conversely, in VLR based GS schemes such as ours, trusted authorities are assumed to deanonymize (open) signatures only when the signer is suspected to be malicious. In [32], Emura *et al.* proposed a light-weight linking based GS scheme with efficient revocation checking. However, during revocation this scheme requires $O(R)$ group operations by the group manager, which restricts the scheme to be used in dynamic crowdsensing-settings where short-lived pseudonyms are critical.

Verifier local revocation. VLR-based GS schemes [16, 22, 26, 48, 49] are known to be more practical than the other schemes [24]. Some VLR-based GS schemes [22, 48, 49] support backward unlinkability. In general, these VLR-based GS schemes need $O(R)$ expensive operations to do revocation checking. The authors in [11] presented a new GS scheme with probabilistic revocation (GSPR) that drastically improves the performance of revocation check, compared to the prior art. However, probabilistic revocation checking resulting in false positives (*i.e.*, valid signatures mistaken as generated by revoked participants) may not be desirable in crowdsensing. Moreover, the experimental evaluation suggests that, revocation check mechanism of SRBE runs faster than GSPR.

Revocation in standard models. There are several standard model constructions of GS schemes based on Groth-Sahai proof system [50], those support constant revocation check. Libert, Peters and Yung (LPY) [51] proposed a construction based on broadcast encryption techniques to support constant revocation check. However, the signature size (96 group elements) and membership certificate size ($O(\log_2 N)$) are extremely large in [51]. In [52], LPY reduced the membership certificate size to constant, with the increased cost of public key size ($O(\log_2 N)$) and signature size (144 group elements). Attrapadung *et al.* [53] proposed another scheme to reduce the size of revocation list to constant. However, the signature size (98 group elements) of this scheme is still large. Recently, Ohara *et al.* [54] proposed a new GS scheme to retain the constant revocation check complexity like [51] with a shorter signature size in ROM. Unfortunately, like the original scheme [51], this scheme also has a large membership certificate size ($O(\log_2 N)$). Moreover, the revocation complexity of all of these schemes is at least $O(R)$ [53]. Most importantly, all of these group signature schemes maintain the revocation list at the signers' end. Thus overall applicability of these schemes in crowdsensing applications envisioned in this paper remains questionable, where light-weight solutions are desirable. It is still open to use the techniques of standard model schemes in ROM to achieve a practical GS scheme with constant revocation checking.

Other revocation techniques. Expensive revocation check has been a major performance bottleneck for anonymous credential schemes as well. In [55, 56], authors proposed an efficient VLR mechanism for anonymous credential systems supporting backward unlinkability. To generate and distribute the revocation list for an epoch, it requires $O(R)$ exponentiation operations of large numbers (expensive) at the revocation authority's end. Like linking-based schemes [21], revocation scheme proposed in [56] requires an online central OCSP server to check the revocation status of signatures from all the verifiers, which introduces additional problems, such as extra communication overheads and the surveillance capability of the OCSP server. Camenisch *et al.* [46] presents a new revocation scheme for anonymous credentials based on n-times unlinkable proofs construction, which overcomes previously mentioned performance overhead. However, it does not support backward unlinkability. As a result, after the revocation of a user device due to legitimate causes (*e.g.*, lost or stolen), all the proofs produced by the device become linkable. The size of the revocation token per user is also linear with the total number of pseudonyms, which makes it challenging to use short epochs. On the other hand, unlike [55, 56], SRBE does not require centralized computations for revocation management and unlike [46], SRBE supports backward unlinkability and constant sized revocation tokens.

2.2 Privacy in CrowdSensing

The privacy concerns in crowdsensing were first pointed out in [57], immediately followed by [58]. AnonySense [28], a privacy preserving crowdsensing framework offers strong privacy protection at the data collection server. AnonySense was one of the earliest solutions that utilize group signatures for crowdsensing. As pointed in [59], the way AnonySense [28] employs group signatures renders it vulnerable to Sybil attacks [60]. Because in AnonySense, it is impossible to identify signatures from the same participant, without opening the signatures of *all* data reports. As a result, misbehavior detection becomes a lengthy and inefficient process, also requiring the de-anonymization of benign reports.

It is conceivable that the inherent openness of privacy preserving systems exposes itself to abuse. Therefore, it is important to efficiently identify abusive users. SPPEAR [27] and SPPEAR with enhanced incentive provisioning [59] are focused on both anonymity and accountability. In SPPEAR [27], BU-anonymity is achieved through pseudonym-based signature approach. However, because of using the pseudonym based signature scheme to share data, SPPEAR incorporates extra public-key certificate management overhead (*e.g.*, pseudonym certificate generation, acquisition, distribution, revocation), which affects the scalability and performance of the system. In contrast, GROUPSENSE provides an alternative approach to solve the problem and also spares the requirement of such public-key certificate management overhead.

3 Motivation and Definitions

We give some intuitions to our design in Section 3.1, define the operations of SRBE in Section 3.2 and formally define the security of SRBE in Section 3.3.

3.1 Motivation

To motivate our design, we describe several straightforward schemes that naively extend a secure group signature scheme with short-lived pseudonyms. Using these schemes, we demonstrate the challenges to preserve security requirements while building an efficient scheme with fast $O(\log_2 R)$ revocation check (where R is the total number of revoked users) without compromising the security.

Failed Scheme I: Consider a naive scheme, where an existing group signature scheme is modified as follows. In addition with private key parameters, each signer (*i.e.*, crowdsensing participant) is assigned with a set of T short-lived pseudonyms p_js by the group manager, where $j \in [1, T]$. When submitting sensory data, the signer concatenates the message m with her pseudonym p_j, and signs $(m \| p)$ following the adopted group signature scheme. The data and signature are submitted in an end-to-end secure channel between the participant and the data-collection server (*e.g.*, HTTPS). The verifier can use a binary search tree to maintain revoked pseudonyms. Thus, revocation check can be done efficiently.

However, this scheme is not secure. Any verifier can learn the pseudonym of the signer. If the verifier is also a member of the group, then she can forge signatures using others' pseudonyms, which violates the *traceability property* (Definition 3.4). Thus, it is necessary to have an easily verifiable correspondence between pseudonyms and the private keys.

Failed Scheme II: Let's assume, we repair Scheme I to preserve the traceability property. Now to revoke a signer the group manager needs to send all the corresponding pseudonyms to the verifier. Thus, the size of revocation token to be transmitted from the group manager to the verifier becomes $O(T)$, where T is the total number of pseudonyms. To overcome this problem, one may generate pseudonym p_j for time interval j as follows:

$$p_j = H_z^j(SEED) \quad \forall j \in [1, T] \tag{1}$$

However, using Equation 1, a verifier can link all the pseudonyms corresponding to a signer, even if the signer is not revoked. Hence, the anonymity is compromised.

Intuitions about SRBE. By observing the failed schemes, we see that using pseudonyms to achieve sublinear revocation check for a GS scheme is not straightforward. Here, we provide a high-level overview of the intuitions behind the design choices of SRBE.

Fixing Scheme I: In SRBE, pseudonyms are called pseudoIDs (PID_{ij} represents the pseudoID of signer i at time epoch j). Our SRBE embeds pseudoIDs into the secret keys for the signers, so that nobody other than the honest signer can "claim" the ownership (See, `Join` protocol in Section 4.1 for details). Such embedding satisfies the following properties.

1. Signers are restricted to use issued pseudoIDs only.
2. Signer i is restricted to use PID_{ij} for time period j.
3. Even if one knows PID_{ij}, she cannot forge signatures. The reconstruction of signing key is not feasible even with the knowledge of PID_{ij}s.

Fixing Scheme II: In SRBE, the pseudoIDs at a given time are generated using a combination of a hash chain and a reverse hash chain (See, Step 6 in `Join` protocol in Section 4.1), so that the revocation token size becomes constant without compromising security.

The salient features of SRBE are summarized as follows.

1. Embedding of pseudoIDs in private key parameters and tying a pseudoID to an epoch (provable under the assumptions similar to Boneh-Boyen full signature scheme[61]).
2. PseudoID generation uses a combination of a hash chain and a reverse hash chain to maintain BU-anonymity with revocation efficiency.
3. The use of pseudoIDs as revocation token enables verifiers to store revocation tokens in standard data-structures for efficient look up.

These new features lead to new capabilities (listed below), which have not previously been realized in the GS literature.

1. VLR based *sublinear* revocation and revocation checking mechanism.
2. BU-anonymity with *constant* revocation token size per signer.

Trade-off. The limitation of using same pseudoID by a signer to produce signatures for a given time period is that, it makes the signatures linkable within that time period. However, signatures from different time intervals are unlinkable. It still remains an open problem to design a group signature scheme with sublinear revocation and unlinkability support within a time interval and across intervals.
The significance of our SRBE scheme is that it has the potential to make large-scale smartphone applications, whose privacy costs were previously formidable, become a reality. It provides a practical fast alternative to existing group signatures, through leveraging the tradeoff between unlinkability and interval duration.

3.2 Definitions of SRBE Operations

There are three types of roles: Group Manager (GM), User or Signer and Verifier. The SRBE scheme consists of the following algorithms:

- **`KeyGen(`1^λ`):`** The GM runs this algorithm, that takes security parameter λ as input and outputs a group public key gpk, a group manager's secret key gms and initializes a registration list reg.
- **`Join:`** This is an interactive protocol between GM and the user i to add user i as a member of the group. On successful execution, the user i obtains the secret key gsk_i, the GM updates reg with an entry reg_i and gets revocation token list $grt_i = \{grt_{ik}\}, \forall k \in [1, T]$, where grt_{ik} is the revocation token of user i at time period k. The revocation token is used in **`Revoke`**.
- **`Sign(`gpk, j, gsk_i, M`):`** With gpk, time period j and gsk_i as input, a signer generates signature σ on message M.
- **`Verify(`gpk, j, RL_j, σ, M`):`** This algorithm is run by the verifiers. If both of the following sub-algorithms output the value `valid`, this algorithm outputs the value `valid`; otherwise, it outputs the value `invalid`.
  - **`SignCheck(`gpk, j, σ, M`):`** With gpk, this sub-algorithm outputs the value `valid`, if σ is an hon-

est signature on message M; otherwise, it outputs the value `invalid`.

  - **`RevCheck`**(j, RL_j, σ)**:** This sub-algorithm outputs the value `valid`, if the revocation handle embedded in signature σ is not revoked; otherwise, it outputs `invalid`.

- **`Revoke`**(j, grt_i)**:** This protocol is executed between the GM and all the verifiers to revoke the membership of user i at time period j. On successful execution, verifiers obtains grt_{ij} and then update their current and future revocation lists ($RL_k, \forall k \in [j, T]$) with corresponding revocation handles generated using grt_{ij}.
- **`Open`**(reg, j, σ, M)**:** Given a valid signature σ on a message M at time period j, created by a signer i, the group manager outputs the signer's identity i.

3.3 Security Definitions

In the security definitions of SRBE, we consider that the total number of signers in the group is N and the total number of time periods is T. The SRBE scheme needs to satisfy the following properties.

Definition 3.1. ***(Signature Correctness):*** *The scheme is correct, if and only if for all (gpk, reg, gsk_i, grt_i) generated by `KeyGen` and `Join` algorithms, every signature generated by signer $i \in [1, T]$ using `Sign` algorithm is flagged as `valid` by `Verify` algorithm in time period $j \in [1, T]$, except when the signer is revoked using `Revoke` algorithm, formally,*

`Verify`$(gpk, j, RL_j,$ `Sign`$(gpk, j, gsk_i, M), M) =$ `valid`, iff, signer i is not revoked at time period j, $\forall i \in [1, N]$ and $\forall j \in [1, T]$.

Definition 3.2. ***(Identity Correctness):*** *The scheme is correct, if and only if for all (gpk, reg, gsk_i, grt_i) generated by `KeyGen` and `Join` algorithms, every signature generated by signer $i \in [1, N]$ using `Sign` algorithm in time period $j \in [1, T]$, `Open` algorithm outputs i, formally,*

Open(reg, j, Sign(gpk, j, gsk_i, M), M) = i, $\forall i \in [1, N]$ and $\forall j \in [1, T]$.

Definition 3.3. ***(BU-anonymity):*** *A group signature scheme is said to satisfy backward unlinkability or BU-anonymity property if the probability of winning the following game is negligibly small for any Probabilistic Polynomial Time (PPT) algorithm $\mathcal{A}$.*

Setup: The challenger $\mathcal{B}$ runs `KeyGen`(1^λ) and `Join`, $\forall i \in [1, N]$. She obtains gpk, gsk_is and reg. She sends gpk to $\mathcal{A}$.

Queries: At the beginning of each period j, $\mathcal{A}$ announces the beginning of j to $\mathcal{B}$, so that they both increment j simultaneously. At any time period $j \in [1, T]$, algorithm $\mathcal{A}$ can make queries to the challenger, as follows.

- **Signing:** Algorithm $\mathcal{A}$ requests a signature on an arbitrary message M for an arbitrary member i. The challenger computes $\sigma \leftarrow Sign(gpk, j, gsk_i, M)$ and returns the signature σ to $\mathcal{A}$.
- **Corruption:** Algorithm $\mathcal{A}$ requests the secret key of signer i. The challenger responds with the key gsk_i.
- **Revocation:** Algorithm $\mathcal{A}$ requests the revocation token of the signer i at time interval j. The challenger responds with the revocation token grt_{ij}.

Challenge: Algorithm $\mathcal{A}$ outputs a message M, time period j^* and two signers i_0, i_1, who are neither corrupted nor revoked at time period j^*. Challenger chooses a bit $b \xleftarrow{R} \{0, 1\}$ uniformly at random, computes a signature on M by signer i_b as $\sigma^* \leftarrow Sign(gpk, j^*, gsk_{i_b}, M)$ and provides σ^* to $\mathcal{A}$.

Restricted Queries: After obtaining the challenge, algorithm $\mathcal{A}$ is allowed to make additional queries of the challenger, restricted as follows.

- **Signing:** As before, but if $\mathcal{A}$ issues any signing queries for i_0 and i_1 at time period j^*, $\mathcal{B}$ reports failure and exits.
- **Corruption:** As before, but if $\mathcal{A}$ issues corruption queries for i_0 and i_1 at any period, $\mathcal{B}$ reports failure and exits.
- **Revocation:** As before, but $\mathcal{A}$ can only issue revocation queries for i_0 and i_1 at any period strictly later than j^*.

Output: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. The adversary wins if $b' = b$. We define her advantage in attacking the scheme to be $|Pr[b = b'] - \frac{1}{2}|$.

Definition 3.4. ***(Traceability):*** *We say that the proposed group signature scheme is traceable, if the probability of winning the following game is negligibly small for each PPT algorithm $\mathcal{A}$.*

Setup: The challenger $\mathcal{B}$ runs `KeyGen`(1^λ) and `Join`, $\forall i \in [1, N]$. She obtains gpk, gsk, grt and reg. She sends gpk and grt to $\mathcal{A}$ and sets $U \leftarrow \emptyset$.

Queries: At the beginning of each period j, $\mathcal{A}$ announces the beginning of j to $\mathcal{B}$, so that they both increment j simultaneously. At any time period $j \in [1, T]$, Algorithm $\mathcal{A}$ can issue queries to the challenger, as follows.

- **Signing:** Algorithm $\mathcal{A}$ requests a signature on an arbitrary message M for an arbitrary member i. The challenger computes $\sigma \leftarrow Sign(gpk, j, gsk_i, M)$ and returns the signature σ to $\mathcal{A}$.
- **Corruption:** Algorithm $\mathcal{A}$ requests the secret key of signer i. The challenger sets $U \leftarrow U \cup \{i\}$ responds with the key gsk_i.

Output: Algorithm $\mathcal{A}$ outputs a message M^* and a signature σ^* for time period j^*. $\mathcal{A}$ wins if:

1. `SignCheck`$(gpk, j^*, \sigma^*, M^*)$ return `valid`;

2. σ^* traces to some signer outside of U or the `Open` algorithm fails; and
3. $\mathcal{A}$ did not obtain σ^* by making a signing query on message M^*.

Definition 3.5. ***(Exculpability):*** *A VLR group signature scheme is said to satisfy exculpability property, if no PPT algorithm can forge a signature that can be attributed to an honest (i.e., not corrupted) member such that the member cannot dispute. Formally, the probability of winning the following game is negligibly small for any PPT algorithm $\mathcal{A}$.*

Setup: The challenger runs `KeyGen(`1^λ`)`. She obtains gpk, gms and reg. She stores gpk and sends gpk, gms and reg to $\mathcal{A}$. Challenger initializes a list of revocation lists $\{RL_j\}$ as empty, $\forall j \in [1, T]$.

Queries: At the beginning of each period j, $\mathcal{A}$ announces the beginning of j to $\mathcal{B}$, so that they both increment j simultaneously. At any time period $j \in [1, T]$, algorithm $\mathcal{A}$ can make queries to the challenger, as follows.

– **Join:** Algorithm $\mathcal{A}$ requests the creation of a new signer $i \in [1, N]$ at period j. Challenger performs `Join` protocol as the new user with $\mathcal{A}$ and gets gsk_i, where $\mathcal{A}$ plays the role of the group manager. $\mathcal{A}$ gets a revocation token list grt_i for the member and an entry reg_i to be inserted into the registration list reg.
– **Signing:** Same as BU-anonymity game.
– **Corruption:** Algorithm $\mathcal{A}$ requests the secret key of signer i. The challenger responds with the key gsk_i. The challenger updates its current and future revocation lists (RL_k, $\forall k \in [j, T]$) with corresponding revocation handles generated using grt_{ij}.

Challenge: Algorithm $\mathcal{A}$ outputs a message M^*, time period j^*, a signature σ^* and a signer i^*. We say that $\mathcal{A}$ wins the game if all the following statements hold:

1. $\mathcal{A}$ did not obtain σ^* from signing query on M^*.
2. `SignCheck(`gpk, j^*, σ^*, M^*`)` return `valid`.
3. `Open(`reg, j^*, σ^*, M^*`)` $= i^*$.
4. $\mathcal{A}$ did not corrupt signer i^*.
5. The challenger cannot dispute the knowledge of signer i^*'s secret key gsk_{i^*} such that $\mathcal{A}$ did not obtain σ^* using gsk_{i^*}.

The Condition 5 was formalized in [62]. Note that, like other standard group signature schemes, this exculpability game assumes honest execution of `KeyGen(`1^λ`)`. However, the exculpability guarantee without such assumption is still open.

4 SRBE Construction

Our SRBE group signature scheme is based on bilinear map which is one of the most widely used mathematical tool to build numerous cryptographic schemes (*e.g.*, signatures [63], aggregate signatures [64], group signatures [16], role-based signatures [65], identity-based encryption [66–68], etc.). Our security and anonymity guarantees rely on several cryptographic assumptions, including the Decision Linear (DLIN) assumption [69], the Discrete Logarithm (DL) assumption, and q-Bilinear Strong Diffie-Hellman (BSDH) assumption [25]. They are defined next.

Definition 4.1. *Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three multiplicative cyclic group of prime order p. g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. ψ is an isomorphism between $\mathbb{G}_2$ and $\mathbb{G}_1$ with efficiently computable homomorphism in both directions. We say e is a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ with the following properties:*

1. *Bilinear: for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$, $e(u^a, v^b) = e(u, v)^{ab}$.*
2. *Non-degenerate: $e(g_1, g_2) \neq 1$.*

Definition 4.2. *DLIN Problem in $\mathbb{G}$ is defined as follows. Given u, v, h, u^a v^b, $z \in \mathbb{G}$, where $a, b \in \mathbb{Z}_p^*$, as inputs, one needs to output* yes*, if $z = h^{a+b}$, or to output* no*, if $z \xleftarrow{R} \mathbb{G}$.*

We say that (t, ϵ)-DLIN assumption holds in $\mathbb{G}$, if no polynomial t-time algorithm has an advantage of at least ϵ at solving DLIN problem in $\mathbb{G}$.

Definition 4.3. *q-BSDH Problem in $(\mathbb{G}_1, \mathbb{G}_2)$ is defined as follows. Given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, \cdots, g_2^{\gamma^q})$ as inputs, the problem is to output a pair $(e(g_1, g_2)^{\frac{1}{\gamma+x}}, x)$, where $x \in \mathbb{Z}_p^*$, $g_2 \xleftarrow{R} \mathbb{G}_2$, $g_1 = \psi(g_2)$, and $\gamma \xleftarrow{R} \mathbb{Z}_p^*$.*

We say that (q, ϵ)-BSDH assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, if no polynomial t-time algorithm has an advantage of at least ϵ at solving BSDH problem in $(\mathbb{G}_1, \mathbb{G}_2)$.

Definition 4.4. *DL Problem in $\mathbb{G}_1$ is defined as follows. Given $g, g^a \in \mathbb{G}_1^2$, where $a \in \mathbb{Z}_p^*$, as inputs, output a.*

We say that (t, ϵ)-DL assumption holds in $\mathbb{G}_1$, if no polynomial t-time algorithm has an advantage of at least ϵ at solving DL problem in $\mathbb{G}_1$.

SRBE scheme also uses $H_z : \{0,1\}^* \longrightarrow \mathbb{Z}_p^*$ and $H_g : \{0,1\}^* \longrightarrow \mathbb{G}_2^2$ [70] as collision resistant hash functions treated as random oracles, where H_z, H_g are considered to be public knowledge.

Note that, the bilinear map we use here is of Type-I, which is necessary for an efficient instantiation of H_g [71]. If $\mathbb{G}_1 = \mathbb{G}_2$, then ψ is an identity map, which is trivial to calculate. By considering more general case of Type-I bilinear map, where $\mathbb{G}_1 \neq \mathbb{G}_2$ but there exists efficiently computable bilinear map e and isomorphism ψ, we can take advantage of certain families of non-supersingular elliptic curves (*e.g.*, MNT [72]) to obtain short signatures. As shown in [69], our security assumptions hold for generic bilinear maps, including $\mathbb{G}_1 = \mathbb{G}_2$ or $\mathbb{G}_1 \neq \mathbb{G}_2$ with efficiently computable e and ψ.

4.1 SRBE Scheme

In this section, we present our SRBE group signature scheme, which extends the classic VLR-based group signature scheme by Boneh and Shacham [16]. SRBE stands for *sublinear revocation with backward unlinkability and exculpability*. We define, $\tau_j = H_z(j), \forall j \in [1, T]$.

`KeyGen`(1^λ): For the given security parameter $\lambda \in \mathbb{N}$, this algorithm chooses a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$, with λ-bit prime order p and isomorphism ψ. Then it generates the group public key gpk and the group manager's secret gms through the following steps.

1. Select a generator $g_2 \xleftarrow{R} \mathbb{G}_2$ and set $g_1 = \psi(g_2)$ such that g_1 is a generator of $\mathbb{G}_1$.
2. Select $\gamma_1, \gamma_2 \xleftarrow{R} \mathbb{Z}_p^*$ and compute $w_1 = g_2^{\gamma_1}$, $w_2 = g_2^{\gamma_2}$.

The group public key is defined as $gpk = (g_1, g_2, w_1, w_2)$ and the group manager's secret key is defined as $gms = (\gamma_1, \gamma_2)$. Finally, the algorithm sets the registration list reg to empty and outputs (gpk, gms). Note that, only group manager has the access to the registration list reg.

`Join`: The interactive protocol is performed securely between the group manager (GM) and a new user i. Steps 6, 7 and 8 are the most important steps of join protocol. Step 6 generates pseudoIDs to ensure BU-anonymity and also enables the scheme to have constant sized revocation token. Steps 7 and 8 embed pseudoIDs in the secret parameters by preserving identity correctness.

1. GM sends a nonce $n_i \xleftarrow{R} \mathbb{Z}_p^*$ to the User.
2. User selects $f_i \xleftarrow{R} \mathbb{Z}_p^*$ and sets $F_i = g_1^{\frac{1}{f_i}}$. User chooses $r_f \xleftarrow{R} \mathbb{Z}_p^*$ and computes $R = g_1^{r_f}$. User also computes $c = H_z(gpk, F_i, R, n_i)$ and $s_f = r_f + \frac{c}{f_i}$. User reselects f_i, in the most unlikely case when $\frac{c}{f_i} = 1$.
3. User sends (F_i, c, s_f) to GM.
4. GM computes $R' = g_1^{s_f} F_i^{-c}$ and checks that $s_f \xleftarrow{R} \mathbb{Z}_p^*$ and $c = H_z(gpk, F_i, R', n_i)$.
5. GM selects two secret seeds $SEED_{i1}, SEED_{i2} \xleftarrow{R} \mathbb{Z}_p^*$.
6. GM generates $PID_{ij}, \forall j \in [1, T]$ using the following equation (Equation 2) and then sets the list of pseudoIDs $PID_i = \{PID_{ij}\}, \forall j \in [1, T]$.

$$
\begin{aligned}
HC_j &= H_z^j(SEED_{i1}) \\
RHC_{T+1-j} &= H_z^{T+1-j}(SEED_{i2}) \\
PID_{ij} &= H_z(HC_j \oplus RHC_{T+1-j})
\end{aligned}
\tag{2}
$$

Here $H_z^x(.)$ means applying the hash function H_z for x times. (The mechanism of using multiple cryptographic hash chains was also employed in [73] to protect user privacy in location-based systems.)

7. GM computes $\pi_i = \Pi_{j=1}^T(\gamma_1 + \gamma_2\tau_j + PID_{ij}), \forall j \in [1, T]$. Then computes, $A_i = F_i^{\frac{1}{\pi_i}}$ and $B_i' = g_2^{\pi_i}$.
8. GM computes, $C_{ij}' = g_2^{\pi_i/(\gamma_1+\gamma_2\tau_j+PID_{ij})}, \forall j \in [1, T]$. In the most unlikely case, if $\pi_i = 0$, restart from step 1.
9. GM defines $gsk_i' = (SEED_{i1}, SEED_{i2}, A_i, B_i', \{C_{ij}'\}), \forall j \in [1, T]$ and sends gsk_i' to user.
10. Using gsk_i', user calculates $B_i = B_i'^{f_i}$ and $C_{ij} = C_{ij}'^{f_i}$, $\forall j \in [1, T]$ and stores them.
11. Using gsk_i', user also calculates $PID_{ij}, \forall j \in [1, T]$ as before and verifies $e(A_i, B_i) = e(g_1, g_2)$ and $e(g_1, B_i) = e(\psi(w_1)\psi(w_2^{\tau_j})g_1^{PID_{ij}}, C_{ij}), \forall j \in [1, T]$.
12. On successful verification, user stores the secret key gsk_i = (f_i, $SEED_{i1}$, $SEED_{i2}$, A_i, B_i, $\{C_{ij}\}$), $\forall j \in [1, T]$, otherwise discards them and outputs error.

On successful execution, the user i gets the secret key gsk_i = (f_i, $SEED_{i1}$, $SEED_{i2}$, A_i, B_i, $\{C_{ij}\}$), $\forall j \in [1, T]$. The GM updates reg with an entry $reg_i = (F_i, PID_i)$ and gets revocation token list $grt_i = \{grt_{ij}\}$, where $grt_{ij} = (H_z^j(SEED_{i1}), SEED_{i2}), \forall j \in [1, T]$. The GM erases the intermediate hash values HCs and RHCs.

`Sign`(gpk, j, gsk_i, M): The inputs to the signing algorithm include the group public key gpk, time period j, the signer's secret key gsk_i, and the message to be signed $M \in \{0, 1\}^*$. This algorithm generates a signature σ on M using the following steps.

1. Compute PID_{ij} as before.
2. To generate a signature in the time period j, use $(A_i, B_i, C_{ij}, PID_{ij})$ as the credentials for signing. After this time interval, discard the PID_{ij}. When all the pseudoIDs are exhausted, group manager should run the *Join* algorithm again to generate new set of secret keys and pseudoIDs (gsk_i) for the user.
3. Select $r \xleftarrow{R} \mathbb{Z}_p^*$ Compute $(\hat{u}, \hat{v}) = H_g(gpk, r, M, PID_{ij})$. Also calculate their images in $\mathbb{G}_1$, set $u = \psi(\hat{u})$, $v = \psi(\hat{v})$.
4. Select $\alpha, \beta, \delta \xleftarrow{R} \mathbb{Z}_p^*$, and compute $T_1 = u^\alpha, T_2 = A_i v^\alpha, T_3 = B_i^\beta$ and $T_4 = C_{ij}^\delta$.
5. Compute the signature of knowledge (SPK), V which is expressed as follows.

$$\begin{aligned}
V &= SPK\{(\alpha, \beta, \delta, A_i, B_i, C_{ij}) : T_1 = u^{\alpha} \\
&\wedge T_2 = A_i v^{\alpha} \wedge T_3 = B_i^{\beta} \wedge T_4 = C_{ij}^{\delta} \\
&\wedge e(A_i, B_i) = e(g_1, g_2) \\
&\wedge e(g_1, B_i) = e(g_1^{\gamma_1} g_1^{\gamma_2 \tau_j} g_1^{PID_{ij}}, C_{ij})\}(M) \\
&= SPK\{(\alpha, \beta, \delta, A_i, B_i, C_{ij}) : T_1 = u^{\alpha} \\
&\wedge e(T_2, T_3) = e(v, T_3)^{\alpha} e(g_1, g_2)^{\beta} \\
&\wedge 1 = e(g_1, T_3)^{\delta} e(\psi(w_1)\psi(w_2^{\tau_j}) g_1^{PID_{ij}}, T_4)^{-\beta}\}(M)
\end{aligned}$$

This SPK is computed with the following steps.

(a) Select blinding factors $r_\alpha, r_\beta, r_\delta \xleftarrow{R} \mathbb{Z}_p^*$, and compute

$$R_1 = u^{r_\alpha}, R_2 = e(v, T_3)^{r_\alpha} e(g_1, g_2)^{r_\beta},$$
$$R_3 = e(g_1, T_3)^{r_\delta} e(\psi(w_1)\psi(w_2^{\tau_j}) g_1^{PID_{ij}}, T_4)^{-r_\beta}.$$

(b) Compute the challenge c as $c = H_z(gpk,\ M,\ j,\ PID_{ij},\ T_1, T_2, T_3, T_4, R_1, R_2, R_3)$.

(c) Compute responses, $s_\alpha = r_\alpha + c\alpha$, $s_\beta = r_\beta + c\beta$, and $s_\delta = r_\delta + c\delta$.

The output of this algorithm is the signature $\sigma = (r,\ PID_{ij},\ T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_\delta)$.

`Verify(`gpk, j, RL_j, σ, M`):` The verification algorithm takes the group public key gpk, the revocation list RL_j at time period j, the signature σ, and the message M as input. Using the following sub-algorithms, it verifies two things: (1) whether the signature was honestly generated, and (2) revocation status of the T_5, embedded in σ. If both the sub-algorithms output `valid`, this algorithm outputs `valid`; otherwise it outputs `invalid`.

(a) **`SignCheck(`gpk, j, σ, M`):`** With the group public key gpk and a signature σ on a message M, this sub-algorithm outputs $valid$ if σ is a valid signature on M as follows.
   1. Compute $(\hat{u}, \hat{v}) = H_g(gpk, r, M, PID_{ij})$ and calculate their images in $\mathbb{G}_1$, like, $u = \psi(\hat{u})$ and $v = \psi(\hat{v})$.
   2. Retrieve:

$$\tilde{R_1} = u^{s_\alpha} T_1^{-c}, \tilde{R_2} = e(v, T_3)^{s_\alpha} e(g_1, g_2)^{s_\beta} e(T_2, T_3)^{-c}$$
$$\tilde{R_3} = e(g_1, T_3)^{s_\delta} e(\psi(w_1)\psi(w_2^{\tau_j}) g_1^{PID_{ij}}, T_4)^{-s_\beta}.$$

   3. Check the correctness of the challenge c as

$$c \stackrel{?}{=} H_z(gpk, M, j, PID_{ij}, T_1, T_2, T_3, T_4, \tilde{R_1}, \tilde{R_2}, \tilde{R_3}).$$

   If the above equation holds, this sub-algorithm outputs `valid`; otherwise, it outputs `invalid`.

(b) **`RevCheck(`j, RL_j, σ`):`** The inputs to the revocation check algorithm are the PID_{ij} embedded in the signature σ and the revocation list RL_j. The purpose of this sub-algorithm is to check whether the PID_{ij} exists in RL_j. The checking can be accomplished by running a fast *binary search* in RL_j.

`Revoke(`j, grt_i`):` GM initiates this protocol by broadcasting the revocation token grt_{ij} from signer i's revocation token list grt_i at time period j to the verifiers, if the membership of the signer is needed to be revoked. Upon receiving it, the verifiers calculate the revoked users' current and future pseudoIDs using Equation 2 and update its revocation lists $RL_k, \forall k \in [j, T]$ by inserting the pseudoID, PID_{ik} in a sorted order.

`Open(`reg, j, σ, M`):` With the valid signature σ on message M, the actual signer of the signature is identified using the following steps.

1. Search the registration list reg for the signer i, who generated the signature σ with the pseudoID, PID_{ij} at time period j.
2. If a match is successfully found, outputs i; otherwise, outputs 0 to indicate a failure.

4.2 Security Analysis

It can be shown that SRBE satisfies the signature correctness and the identity correctness properties, by constructing the frameworks discussed in [16]. Here, we prove the BU-anonymity (Theorem 4.5), traceability (Theorem 4.6) and exculpability (Theorem 4.7) properties of SRBE under DLIN assumption, BSDH assumption and DH assumption respectively. Proofs are provided in Appendix A.

Theorem 4.5. ***(BU-Anonymity).*** *In the random oracle model, suppose an algorithm $\mathcal{A}$ breaks the BU-anonymity of* SRBE *scheme with an advantage of ϵ after q_H hash queries and q_S signing queries, then there exists an algorithm $\mathcal{B}$ that breaks the DLIN assumption with an advantage of $\frac{\epsilon}{2}(\frac{1}{N^2} - \frac{q_S q_H}{p})$.*

Theorem 4.6. ***(Traceability).*** *In a random oracle model, suppose an algorithm $\mathcal{A}$ breaks the traceability of* SRBE *with an advantage of ϵ after q_H hash queries, then there exists an algorithm $\mathcal{B}$ that breaks the q-BSDH assumption with an advantage of $(\epsilon/N - 1/p)^2/(16q_H)$, where $q = (N+1)T$.*

Theorem 4.7. ***(Exculpability).*** *In a random oracle model, suppose an algorithm $\mathcal{A}$ breaks the exculpability of* SRBE *with an advantage of ϵ, then there exists an algorithm $\mathcal{B}$ that breaks the DL assumption with non-negligible probability.*

Note that, like other GS schemes in Random Oracle Model (ROM), reductions to the standard assumptions for all these theorems are non-tight.

4.3 Complexity Analysis

Scheme	Function	Exp. in $\mathbb{G}_1/\mathbb{G}_2$	Exp. in $\mathbb{G}_T$	Bilinear Ops.	Big O
SRBE (Ours)	Sign	5	4	3	$O(1)$
	SignCheck	3	5	4	$O(1)$
	RevCheck	0	0	0	$O(\log_2 R)$
	Revoke	0	0	0	$O(\log_2 R)$
CLHZ [48]	Sign	7	5	5	$O(1)$
	SignCheck	7	6	7	$O(1)$
	RevCheck	R	0	0	$O(R)$
	Revoke	0	0	0	$O(1)$
BS [16]	Sign	5	3	3	$O(1)$
	SignCheck	4	4	4	$O(1)$
	RevCheck	0	0	$R+1$	$O(R)$
	Revoke	0	0	0	$O(1)$
BSNSW [26]	Sign	3	1	1	$O(1)$
	SignCheck	0	2	5	$O(1)$
	RevCheck	0	0	$R+2$	$O(R)$
	Revoke	0	0	0	$O(1)$
GSPR [11]	Sign	6	4	3	$O(1)$
	SignCheck	2	5	4	$O(1)$
	RevCheck	0	0	0	$O(1)$
	Revoke	0	0	0	$O(T)$

A. Computational Overhead

We implemented four state-of-the-art group signature schemes, CLHZ [48], BS [16], BCNSW [26], and GSPR [11] for performance comparison, the results of which confirm our fast revocation property and are shown in Figure 2 in Section 6.2, of them only CLHZ supports backward unlinkability. In this section, we theoretically compare the computational and communication overhead of them. Note that all the selected schemes are chosen from VLR based schemes. As explained in 2, VLR based schemes are best suited for crowdsensing applications. GSPR is the only scheme with probabilistic revocation and also all the signatures generated by a signer in an epoch are linkable. In Table 1, we compare the most frequent operations of them in terms of exponentiation, bilinear operations and the overall runtime complexity. In runtime complexity, R indicates the size of the revocation list. For GSPR, T denotes the number of time periods. We consider only the computationally most expensive operations - i.e., exponentiation in $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$ and bilinear operations. Since in our implementation, $\mathbb{G}_1 = \mathbb{G}_2$, the application of isomorphism is not considered here. For both Sign and SignCheck, BCNSW is the most efficient scheme. For SRBE, during signature generation and verification $w_2^{\tau_j}$, $e(g_1, g_2)$ can be precomputed. Some other expensive operations of signing algorithm are also independent of the message, thus further pre-computation is feasible, which we have implemented and the results are presented in Table 3. Although GSPR supports probabilistic revocation, the fast runtime complexity of RevCheck algorithm of both SRBE and GSPR is noticeable. On the other hand, for revoke operation, only SRBE and GSPR have non-constant runtime complexity. However, unlike frequent revocation checking, member revocation is not often. The small increase is justified, as it substantially improves revocation checking from $O(R)$ to $O(\log_2 R)$.

Scheme	Messages	Elem. in $\mathbb{Z}_p^*$	Elem. in $\mathbb{G}_1/\mathbb{G}_2$	Numbers
SRBE (Ours)	Pub. Key	0	4	0
	Priv. key	2	$T+2$	0
	Sign.	6	4	0
	Rev. Token	2	0	0
CLHZ [48]	Pub. Key	0	6	0
	Priv. Key	$T+1$	T	1
	Sign.	5	4	2
	Rev. Token	T	0	0
BS [16]	Pub. Key	0	3	0
	Priv. Key	1	1	0
	Sign.	5	2	0
	Rev. Token	0	1	0
BSNSW [26]	Pub. Key	0	2	0
	Priv. Key	1	3	0
	Sign.	2	3	0
	Rev. Token	0	1	0
GSPR [11]	Pub. Key	0	$T+2$	0
	Priv. Key	1	1	0
	Sign.	5	4	0
	Rev. Token	0	0	1

B. Communication Overhead

Communication costs in terms of various message sizes are shown in Table 2. Here T is the number of time periods. In dynamic crowdsensing environments, the size of signatures and revocation tokens are arguably the most important, because only these two messages are exchanged for multiple times (user sends signatures to the verifier, group manager sends revocation token to multiple verifiers) between entities for a user. The sizes of SRBE's revocation tokens and public keys are much smaller than CLHZ and GSPR, and are comparable to BS and BCNSW schemes. The size of signature for SRBE is shorter than CLHZ scheme and comparable to GSPR, but is higher than BCNSW and BS. Although the exchange of private key parameters is quite infrequent than others, the size of private key is significantly larger for SRBE and CLHZ schemes than the other three schemes. In SRBE, the size growth of private key is due to PID and C values.

One can reduce the space complexity of the private key in SRBE for low storage devices as follows. Only one PID and one C are used in a single time period, so group manager can securely send them to the signing device on demand. On receiving the current values, the device can discard the previous values of PID and C, which can reduce the linear space complexity of the signing device to constant. Adoption of this mechanism will also help to reduce the computational delay due to private key verification during `Join` protocol.

Overall SRBE clearly makes advantageous trade-off between computational and communication overheads. When considering scalability, reducing the computational overhead significantly is much more precious, even at the cost of a slight increase in the communication overhead. In addition to that the backward unlinkability property of SRBE, is also useful in dynamic group settings.

5 SRBE Application in Groupsensing

We define groupsensing to be a controlled crowdsensing scenario where data submission is limited to members of a pre-authorized sensing group. Non-members without proper sensing group credentials cannot submit valid data reports. We show how our SRBE group signature can be applied to realize anonymous-yet-accountable groupsensing.

Our prototype GROUPSENSE is composed of three types of entities: participant's device (PD), data-collection server (DCS), and a trusted group manager (GM) (*i.e.*, the group manager in SRBE). GROUPSENSE allows participants to anonymously sign sensory data and to submit the data to a semi-honest data-collection server. The data collector performs signature verification and revocation checking. However, it is unable to track participants even after the revocation, as data submitted by the same participant are backward unlinkable. The trusted group manager is responsible for credential, revocation management, and possible reward distribution, but even group manager cannot forge signatures for any participants because of the exculpability property of our SRBE scheme. Our experiments in the next section show that GROUPSENSE has the potentials to support massive crowds in practice with fast signature verification coupled with speedy revocation checking.

5.1 Security Model in GroupSense

Threat Model.

We focus on three categories of threats.

- *Data forgery.* Malicious participants may purposely contribute fake data reports (*e.g.*, submit fake traffic congestion reports).
- *Identity forgery.* Unauthorized individuals and devices that are not part of the group may attempt to submit data reports. In addition to that, anyone including the group manager may attempt to forge the identity of a signer to submit malicious/fake data reports.
- *Honest-but-curious data collector.* The data-collection server follows the protocol, but may attempt to track a participant through her data reports. This type adversary is also known as semi-honest. For example, the data-collection server may examine the context and location of sensory data, attempt to pinpoint a participant's IP address history and movement trajectory.

The credential distribution between the group manager and the participants is assumed secure. In addition, we assume that the mobile app on participant's device is trustworthy, *e.g.*, free of spyware, stealth tracking capability, and data-leak vulnerabilities. Advanced collusion and correlation attacks for de-anonymization are out of our scope, *e.g.*, the semi-honest data-collection server colludes with a mobile service provider, or correlates sensory data with known locations of a participant. We assume that external adversaries who may launch disruptive attacks such as DDoS and jamming can be detected with existing solutions. Traffic analysis threats from adversaries that are external to a groupsensing system (*e.g.*, routers, access points, and other network intermediaries) are out of our scope. We explain how anonymous routing (such as TOR) is positioned in GROUPSENSE in the next section.

Security and Privacy Goals.

Under the above attack model, GROUPSENSE has three security and privacy goals: accountability (traceability), identity unforgeability and sensing-time anonymity.

Accountability. The sensing group membership of a misbehaving participant can be identified and revoked efficiently.

Identity Unforgeability. In groupsensing, this goal is two-fold. (1) Data-collection server can verify that received data reports are from valid group members. So that, any data submissions outside of group membership can be automatically discarded. (2) No one including the group manager can forge the identity of a valid signer.

Sensing-time Anonymity. The data reports submitted by a participant do not provide any information that enables the data-collection server to link them with reports of the same participant, even after the signer is revoked.

Collusion and Correlation Attacks.

Although advanced collusion and correlation attacks for de-anonymization are out of our threat model, we briefly describe possible mitigation and open problems. An example of such attacks is where a semi-honest data-collector colludes with a participant's mobile service provider to correlate data submission activities with cell phone activities. Another data source useful for correlation attacks is surveillance video of public places and locations and IP addresses.

For the IP addresses from which a participant connects to the Internet, we distinguish two cases: (1) public-place IP address (*e.g.*, Wi-Fi at hotels and restaurants) that multiple participants may have access to, and (2) private-place IP address (*e.g.*, at a private residence), which can be deterministically mapped to individuals. In the latter case, the data collector can easily link multiple data submissions from a participant's home. Therefore, anonymous routing (such as TOR) is required for data submission from private residences.

However, in the former case, correlation attacks (*e.g.*, with surveillance video of the location) may enable attackers to link signatures. It is unclear how these advanced correlation attacks are defended. Mobile performance of TOR onion routing also needs evaluation in this specific context.

5.2 GroupSense Operations

GROUPSENSE is a crowdsensing prototype that supports anonymity and accountability through SRBE group signature scheme. Key operations in GROUPSENSE are shown in Figure 1 and are built on SRBE operations. We describe the operations in GROUPSENSE below.

Initialization and Recruitment: DCS initiates a crowdsensing campaign by sending a group setup request to the group manager (GM) (step 1). GM divides the entire data collection period into T time epochs. GM invokes `KeyGen` (1^{λ}) function to get (gms, gpk), stores gms secretly and distributes gpk to data-collection server (DCS). During this phase, the DCS specifies the desired sensing tasks including the sensor readings of interest, time period and geographic area to sense. It may also specify the task budget and incentive scheme [74]. In this phase, DCS and GM also agrees on data exchange and communication protocols for their future interactions.

For a particular crowdsensing campaign, the GM is responsible for advertising the task and recruiting participants. Interested participants start `Join` protocol with GM to join a campaign. After successful completion of `Join` protocol, GM obtains (grt_i, reg_i) and participant's device (PD) obtains (gpk, gsk_i) with the information of the DCS. We assume that the communication between PD and GM during `Join` protocol is secured using end-to-end encryption.

Data Collection: Participants perform the sensing task and collect sensory data using PD. PD signs each data report using `Sign` (gpk, j, gsk_i, M) and sends data along with the signature (σ) (step 3) to DCS. Then DCS verifies the signature by invoking `Verify` $(gpk, j, \ RL_j, \sigma, M)$. The DCS server is responsible for storing and processing the collected data, including data aggregation and false data detection [75]. On each data submission, DCS responds with a receipt (signed acknowledgement) to the PD.

Revocation: After detection of a misbehaving participant (*e.g.*, data, submitted by the participant deviates from the normal pattern), the DCS sends the corresponding signature (σ) of that participant to the GM. After receiving the signature, GM opens it to get the identity of the participant by invoking `Open` (reg, j, σ, M). Consequently, the GM executes `Revoke` (j, grt_i) protocol to send the revocation token grt_{ij} back to the DCS (step 4).

Reward Distribution: Metrics for distributing rewards may depend on applications. In general, GM is in-charge for the incentive distribution of GROUPSENSE. If reward distribution demands the assessment of each participant's contribution, PD can submit receipts corresponding to its data submissions to GM.

Security Analysis.

It is straightforward to show that the security goals of GROUPSENSE are achieved by our SRBE group signature scheme. Accountability, sensing-time anonymity, identity unforgeability are enforced by the traceability, BU-anonymity and exculpability property of SRBE, respectively.

In addition, SRBE makes typical Sybil attacks [60, 76] harder on GROUPSENSE, by restricting participants to use one pseudoID at a given time interval. In Sybil attack [60], a participatory node illegitimately claims multiple identities.

GROUPSENSE can support shorter time periods to enforce stronger unlinkability. When all the private keys stored in participant's device (PD) are exhausted, PD refills its private parameters by initiating Join protocol with GM.

6 Evaluation: SRBE & GroupSense

6.1 Implementation

We implemented all five group signature schemes compared in Tables 1 and 2, namely SRBE (ours), CLHZ [48], BS [16], BCNSW [26], and GSPR [11], in *C* using the *PBC library* [77]. We used "Type A" pairing as internally defined in the library, which is constructed with supersingular elliptic curve $E \equiv y^2 = x^3 + x$ over the field F_q for some prime $q = 3 \mod 4$. As both $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of points $E(F_q)$, this pairing is symmetric. In our implementation, an element in $\mathbb{Z}_p^*$ is denoted by 160 bits and an element in $\mathbb{G}_1$ or $\mathbb{G}_2$ is denoted by 512 bits, which implies that the security strength of all these implementations is comparable to an RSA signature with a modulus size of 1024 bits. For SRBE, CLHZ and GSPR schemes, we assume that the duration of each epoch is one day.

Our GROUPSENSE prototype based on our SRBE group signature scheme consists of (1) an Android mobile app for PD and (2) the server-side programs for data collection server (DCS) and group manager (GM). Server-side applications are implemented in JavaEE platform using the scalable Spring Framework. Both Android application and server-side programs use jPBC library [78] (a Java wrapper of PBC library). The PD application invokes `joinGroup` API of GM application for joining the group and `storeData` API for sending sensor data to DCS. In the DCS application, we implemented two services and exposed corresponding RESTful web service APIs named: (1) `storeData`, to receive and store data from PD after verifying the signature; (2) `revoke`, to receive revocation tokens from GM. In the GM application, we implemented four services and exposed corresponding RESTful web service APIs named: (1) `setupGroup`, to setup and initialize the group; (2) `joinGroup`, for the participants to join the campaign; (3) `requestRevocation`, to receive requests for revocation from DCS; (4) `storeContributionAssessments`, to receive and store contribution assessment reports for incentive provisioning from PD.

`joinGroup` supports both `GET` and `POST` requests. PD initiates the protocol and receives nonce through `GET` request and submits F_i for the credential generation in `POST` request as suggested in the `Join` protocol.

6.2 Evaluation

For performance evaluation, we deployed server-side applications in a Tomcat server, running on an *Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz* machine. For client side evaluation, we used *Nexus 7 CPU 1.51 GHz quad-core Krait 300* and *Nexus 10 CPU 1.7 GHz Dual-core Cortex-A15* with Android version 4.4.2. We simulated the real-world environments with a load-testing tool named Gatling[1] while evaluating server-side performance and present the results in boxes showing the inter quartiles (*i.e.*, 25^{th} and 75^{th} percentiles); the line inside the box depicts the average value; and the whiskers show the minimum and maximum values.

Our performance evaluation of SRBE and GROUPSENSE aims to answer the following questions:

– How long does revocation checking take under thousands of revoked users in 5 group signature schemes? (Section 6.2.1)
– How long does signing take on Android devices? What code optimization can be done? (Section 6.2.2)
– How long does Join protocol take overall (both in Android and GM server)? What can be done to minimize the overall delay effect? (Section 6.2.3)
– How does GROUPSENSE data-collection server perform during data submissions under stress testing? (Section 6.2.4)
– How does GROUPSENSE GM server perform during device revocation under stress testing? (Section 6.2.5)

1 http://gatling.io/

6.2.1 Scalability of Revocation Checking

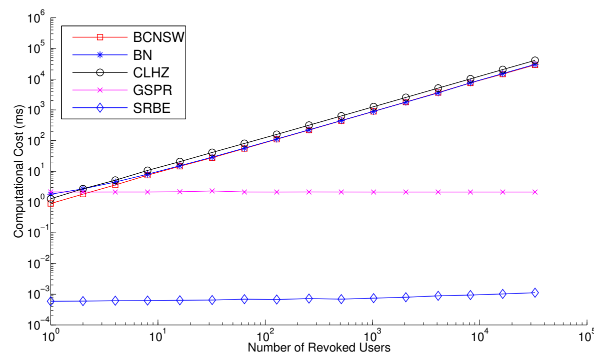


Fig. 2. RevocationCheck runtime with increasing number of revoked users in five group signature schemes.

Figure 2 shows the run time of `RevocationCheck` algorithm in all five group signature schemes under a large number of revoked users. The measurements were obtained by averaging over 1,000 runs of each scheme. The experimental results show that SRBE's RevocationCheck with the binary search tree is significantly faster than others as expected. From Figure 2, we see that GSPR's runtime complexity does not directly depend on number of revoked users. However, it linearly depends on the number of iterations and the size of its piecewise-orthogonal-codes. We used 20 bit long piecewise-orthogonal-codes and the number of iterations was 1. With the linear increase in either of these parameters, the false positive rate decays exponentially, but the computational complexity also increases linearly. So it is conceivable that with constant negligible false positive rate, GSPR's computational complexity will be substantially increased. Hence, it cannot cannot outperform SRBE.

6.2.2 Android Signing Performance

In Table 3, we show the average signing delay of 20 runs in two Android devices (*Nexus 7* and *Nexus 10*) of SRBE, BS [16] and CLHZ [48]. We see that, relative performance of these schemes is consistent with the theoretical comparison in Table 1. We also measured the signing delay by pre-computing the message independent expensive operations for SRBE. The

Table 3. Average signing delays on two different Android devices.

	Nexus 7	Nexus 10
SRBE	2.421s	2.385s
BS [16]	2.189s	2.120s
CLHZ [48]	3.082s	2.787s

precomputed version took on average of 1.806s in *Nexus 7* and 1.686s in *Nexus 10*. Doing precomputations in elliptic curve cryptosystems [79] and group signature [80] are very common to speed up the signing performance.

6.2.3 Join Protocol Performance

In join protocol `joinGroup` service, GM registers the PD and generates the secret parameters for PD. After receiving the parameters, PD verifies and stores them.

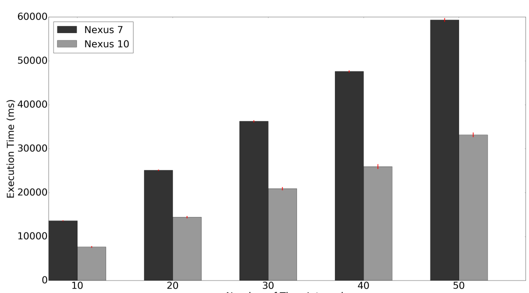


Fig. 3. Computational delay of cryptographic operations during join protocol in android devices.

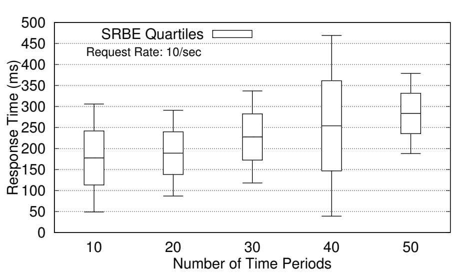
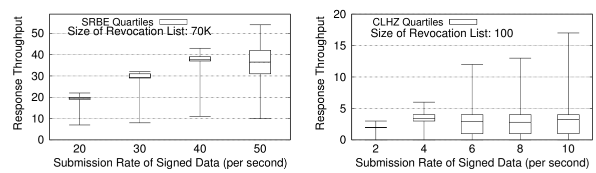


Fig. 4. Number of time periods vs. response time Quartiles of `POST` requests to `joinGroup` service.

In Figures 3 4, we show the impact of number of time intervals on both PD and GM while performing join protocol. In Figure 3, we report the averages of 20 runs. Note that, the performance evaluation shows that the computational overhead of `joinGroup` service does not depend on the size of the registration list or the revocation list.

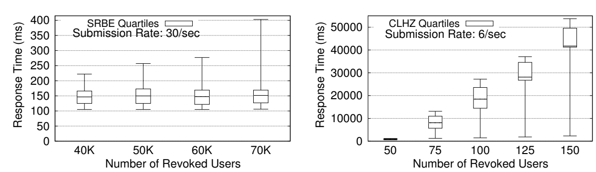
The result depicts the linear increase of both the computational delay and response time with the number of time intervals. Still we observe that, to acquire pseudoIDs for 50 time intervals, overall it takes less than 6s in Nexus 7 and around 4s in Nexus 10, which is faster than the state-of-the art privacy preserving crowdsensing systems (*e.g.*, SPPEAR [27]). Note that, the effect of the computational delay in Android devices can be minimized, if GM sends private parameters on demand as mentioned in Section 4.3.

6.2.4 Data-collection Server Performance



(a) SRBE scheme **(b)** CLHZ scheme

Fig. 5. Data submission rate vs. Throughput Quartiles of `storeData` service.

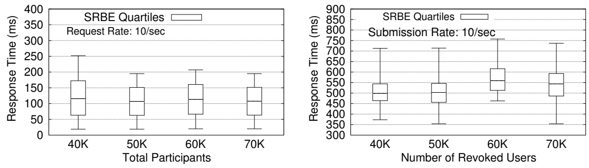


(a) SRBE scheme **(b)** CLHZ scheme

Fig. 6. Revocation list size vs. Response time Quartiles of `storeData` service.

We test the scalability of two implementations of the DCS (specifically the `storeData` service), one with our SRBE and one with CHLZ [48] (baseline). We choose CHLZ as the baseline, because it supports deterministic revocation (*i.e.*, no false alarms), backward unlinkability (signatures are unlinkable, across and within epochs), exculpability and has a revocation complexity like several other schemes. In `storeData` service, DCS verifies the signatures of the submitted data reports and after successful verification it stores data or discards otherwise. It is worth-mentioning that, the overall scalability of DCS server, solely depends on the performance of `storeData` service, where the revocation check is performed and known to be a bottleneck previously. Figure 5a illustrates that, the average throughput with SRBE increases linearly with the data submission rate (before reaching its peak) and Figure 6a illustrates that, the average response time remains constant with the increase of the size of revocation list. We kept both the revocation list size and data submission rate low for CLHZ, as higher values caused server timeout. Here the revocation list size is measured in terms of users.

6.2.5 Revocation Performance



(a) GM **(b)** DCS

Fig. 7. Response time Quartiles for both `requestRevocation` service of GM and `revoke` service of DCS.

After receiving a device revocation request at `requestRevocation` API, GM opens the signature to identify the participant and to find the revocation token of the participant. Then GM sends the revocation token asynchronously to DCS servers by invoking the `revoke` API. After receiving the revocation token, DCS updates the current revocation list instantly and the upcoming revocation lists asynchronously. In Figure 7a and 7b, we observe that the response time to revoke a participant both in GM and DCS end does not increase with the size of registration list (for GM) and revocation list (for DCS). In GM, the average time it takes to revoke a user is around 150ms and in DCS it takes around 550ms to update a revocation list, which is an order of magnitude faster than the prior art (*e.g.*, SPPEAR [27] reported 2.3s (on avg.) for device revocation). As before, here the registration list size and the revocation list size are measured in terms of users. To build the registration list, we considered 100 pseudonyms per user.

6.3 Summary

We summarize our overall performance evaluation below.

(1) On average, SRBE's performance of revocation check of SRBE scheme is 3 order of magnitude greater than the state of the art for a fairly large number of revoked users.

(2) The signing performances of GROUPSENSE with SRBE scheme, in Android devices are comparable to other known group signature schemes. Precomputation of expensive operations gives a fairly better performance gain over the non-precomputed one.

(3) The joining protocol is the most expensive task in GROUPSENSE. Still the overall delay is shorter than the prior art.

(4) The increase in averaged response time of GROUPSENSE data collection procedure is negligible, when we increase the revocation list size from 40K to 70K users. This result is promising, indicating the scalability potentials of GROUPSENSE in practical crowdsensing applications.

(5) The performance of Data collection server (DCS) during device revocation under stress testing is also an order of magnitude greater than the prior art (*e.g.*, SPPEAR [27]).

7 Conclusion

Our work was motivated by the need for supporting large-scale anonymous smartphone applications, such as crowdsensing. Our main technical contribution is a provably secure group signature scheme called SRBE that realizes sublinear revocation checking. Revocation checking is a frequently executed operation required for each signature verification. SRBE also provides typical group signature guarantees including backward unlinkability. Our fast revocation checking is made possible through utilizing and integrating cryptographic, algorithmic, and data structural building blocks. We gave a formal and comprehensive security analysis of SRBE and discussed limitations and trade-offs. Another substantial technical contribution is the SRBE-based crowdsensing prototype with Android support called GROUPSENSE including its security analysis. Our extensive experimental evaluation on GROUPSENSE showed that GROUPSENSE with fast revocation checking scales well with the increase of the revocation tokens.

The significance of our work is that it brings provably secure group signatures closer to deployment in a large scale in practice. Such effort on privacy is necessary with the ever-increasing number of user-centric applications.

8 Acknowledgement

We thank Keita Emura for helpful technical discussion on the anonymity properties. We would also like to thank our shepherd, Aggelos Kiayias, and anonymous reviewers for their help and insightful suggestions. This project has been supported in part by NSF grant CBET-1645121.

References

[1] Mu Lin, Nicholas D. Lane, Mashfiqui Mohammod, Xiaochao Yang, Hong Lu, Giuseppe Cardone, Shahid Ali, Afsaneh Doryab, Ethan Berke, Andrew T. Campbell, and Tanzeem Choudhury. Bewell+: multi-dimensional wellbeing monitoring with community-guided user feedback and energy optimization. In *Wireless Health '12*, pages 1–8, 2012.

[2] Eiman Kanjo. Noisespy: A real-time mobile phone platform for urban noise monitoring and mapping. *Mobile Networks and Applications*, 15(4):562–574, 2010.

[3] Bei Pan, Yu Zheng, David Wilkie, and Cyrus Shahabi. Crowd sensing of traffic anomalies based on human mobility and social media. In *SIGSPATIAL '13*, pages 344–353, 2013.

[4] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[5] Raluca Ada Popa, Andrew J. Blumberg, Hari Balakrishnan, and Frank H. Li. Privacy and accountability for location-based aggregate statistics. In *ACM CCS '11*, pages 653–666, 2011.

[6] Delphine Christin. Privacy in mobile participatory sensing: Current trends and future challenges. *Journal of Systems and Software*, 116:57–68, 2016.

[7] Leyla Kazemi and Cyrus Shahabi. A privacy-aware framework for participatory sensing. *SIGKDD*, 13(1):43–51, 2011.

[8] Facebook urged to tighten privacy settings after harvest of user data. www.theguardian.com/technology/2015/aug/09/facebook-privacy-settings-users-mobile-phone-number, August 2015. [Online; accessed 16-May-2016].

[9] NSA Prism program taps in to user data of Apple, Google and others. http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data, June 2013. [Online; accessed 16-May-2016].

[10] Facebook admits year-long data breach exposed 6 million users. http://www.reuters.com/article/net-us-facebook-security-idUSBRE95K18Y20130621, June 2013. [Online; accessed 16-May-2016].

[11] Vireshwar Kumar, He Li, Jung-Min "Jerry" Park, Kaigui Bian, and Yaling Yang. Group signatures with probabilistic revocation: A computationally-scalable approach for providing privacy-preserving authentication. In *ACM CCS '15*, pages 1334–1345, 2015.

[12] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In *SAC'99*, pages 184–199, 1999.

[13] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[14] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT '91*, pages 257–265, 1991.

[15] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In *EUROCRYPT 2004*, pages 571–589, 2004.

[16] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In *ACM CCS '04*, pages 168–177, 2004.

[17] Maxim Raya and Jean-Pierre Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[18] Xiaodong Lin, Xiaoting Sun, Pin-Han Ho, and Xuemin Shen. GSIS: A secure and privacy-preserving protocol for vehicular communications. *IEEE Trans. Vehicular Technology*, 56(6):3442–3456, 2007.

[19] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, pages 255–270, 2000.

[20] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT '03*, pages 614–629, 2003.

[21] Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Group signatures with linking-based revocation: A pragmatic approach for efficient revocation checks. In *MyCrypt 2016*, 2016. to appear.

[22] Julien Bringer and Alain Patey. Backward unlinkability for a VLR group signature scheme with efficient revocation check. *IACR Cryptology ePrint Archive*, 2011:376, 2011.

[23] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable group signature schemes with constant costs for signing and verifying. In *PKC 2009*, pages 463–480, 2009.

[24] Mark Manulis, Nils Fleischhacker, F Gunther, K Franziskus, and Bertram Poettering. Group signatures: Authentication with privacy. *Bundesamt fur Sicherheit in der Informationstechnik. Tech. Rep*, 2012.

[25] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.

[26] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In *Security and Cryptography for Networks SCN '10*, pages 381–398, 2010.

[27] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. SPPEAR: security & privacy-preserving architecture for participatory-sensing applications. In *WiSec '14*, pages 39–50, 2014.

[28] Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. Anonysense: Privacy-aware people-centric sensing. In *MobiSys '08*, pages 211–224, 2008.

[29] Ioannis Boutsis and Vana Kalogeraki. Privacy preservation for participatory sensing data. In *IEEE Pervasive Computing and Communications (PerCom) '13*, pages 103–113, 2013.

[30] Emiliano De Cristofaro and Claudio Soriente. Extended capabilities for a privacy-enhanced participatory sensing infrastructure (PEPSI). *IEEE Trans. Information Forensics and Security*, 8(12):2021–2033, 2013.

[31] Leyla Kazemi and Cyrus Shahabi. TAPAS: trustworthy privacy-aware participatory sensing. *Knowl. Inf. Syst.*, 37(1):105–128, 2013.

[32] Keita Emura and Takuya Hayashi. A light-weight group signature scheme with time-token dependent linking. In *Lightweight Cryptography for Security and Privacy 2015*, pages 37–57, 2015.

[33] Citizen Science Alliance. http://www.citizensciencealliance.org/. [Online; accessed 04-August-2016].

[34] Seung Geol Choi, Kunsoo Park, and Moti Yung. Short traceable signatures based on bilinear pairings. In *IWSEC 2006*, pages 88–103, 2006.

[35] Vicente Benjumea, Seung Geol Choi, Javier Lopez, and Moti Yung. Fair traceable multi-group signatures. In *Financial Cryptography and Data Security, 2008*, pages 231–246, 2008.

[36] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO '04*, pages 56–72, 2004.

[37] Benoît Libert and Moti Yung. Efficient traceable signatures in the standard model. In *Pairing-Based Cryptography - Pairing*

2009, pages 187–205, 2009.

[38] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In *CRYPTO '15*, pages 296–316, 2015.

[39] Jung Yeon Hwang, Sokjoon Lee, Byung-Ho Chung, Hyun Sook Cho, and DaeHun Nyang. Group signatures with controllable linkability for dynamic membership. *Inf. Sci.*, 222:761–778, 2013.

[40] Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Adding controllable linkability to pairing-based group signatures for free. In *Information Security - 17th International Conference, ISC 2014*, pages 388–400, 2014.

[41] Essam Ghadafi. Efficient distributed tag-based encryption and its application to group signatures with efficient distributed traceability. In *LATINCRYPT 2014*, pages 327–347, 2014.

[42] Toru Nakanishi and Nobuo Funabiki. Efficient revocable group signature schemes using primes. *JIP*, 16:110–121, 2008.

[43] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, pages 61–76, 2002.

[44] Jorn Lapon, Markulf Kohlweiss, Bart De Decker, and Vincent Naessens. Performance analysis of accumulator-based revocation mechanisms. In *Security and Privacy - Silver Linings in the Cloud - 25th IFIP TC-11 International Information Security Conference, SEC 2010, Held as Part of WCC 2010*, pages 289–301, 2010.

[45] Chun-I Fan, Ruei-Hau Hsu, and Mark Manulis. Group signature with constant revocation costs for signers and verifiers. In *Cryptology and Network Security CANS 2011*, pages 214–233, 2011.

[46] Jan Camenisch, Manu Drijvers, and Jan Hajny. Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, WPES '16, pages 123–133, New York, NY, USA, 2016. ACM.

[47] Markulf Kohlweiss and Ian Miers. Accountable metadata-hiding escrow: A group signature case study. *PoPETs*, 2015(2):206–221, 2015.

[48] Cheng-Kang Chu, Joseph K. Liu, Xinyi Huang, and Jianying Zhou. Verifier-local revocation group signatures with time-bound keys. In *ACM ASIACCS '12*, pages 26–27, 2012.

[49] Toru Nakanishi and Nobuo Funabiki. Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. *IEICE Transactions*, 90-A(1):65–74, 2007.

[50] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, pages 415–432, 2008.

[51] Benoît Libert, Thomas Peters, and Moti Yung. Scalable group signatures with revocation. In *EUROCRYPT 2012*, pages 609–627, 2012.

[52] Benoît Libert, Thomas Peters, and Moti Yung. Group signatures with almost-for-free revocation. In *CRYPTO 2012*, pages 571–589, 2012.

[53] Nuttapong Attrapadung, Keita Emura, Goichiro Hanaoka, and Yusuke Sakai. A revocable group signature scheme from identity-based revocation techniques: Achieving constant-size revocation list. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014*, pages 419–437, 2014.

[54] Kazuma Ohara, Keita Emura, Goichiro Hanaoka, Ai Ishida, Kazuo Ohta, and Yusuke Sakai. Shortening the libert-peters-yung revocable group signature scheme by using the random oracle methodology. *IACR Cryptology ePrint Archive*, 2016:477, 2016.

[55] Wouter Lueks, Gergely Alpár, Jaap-Henk Hoepman, and Pim Vullers. Fast revocation of attribute-based credentials for both users and verifiers. In *IFIP '15*, pages 463–478, 2015.

[56] Eric R. Verheul. Practical backward unlinkable revocation in fido, german e-id, idemix and u-prove. *IACR Cryptology ePrint Archive*, 2016:217, 2016.

[57] Katie Shilton, Jeffrey A Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Participatory privacy in urban sensing. In *International Workshop on Mobile Device and Urban Sensing (MODUS)*, 2008.

[58] Apu Kapadia, David Kotz, and Nikos Triandopoulos. Opportunistic sensing: Security challenges for the new paradigm. In *2009 First International Communication Systems and Networks and Workshops*, pages 1–10. IEEE, 2009.

[59] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. Security, privacy & incentive provision for mobile crowd sensing systems. *IEEE IoT*, 2016.

[60] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y. Zhao. Defending against sybil devices in crowdsourced mapping services. In *MobiSys '16*, 2016.

[61] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, 2004.

[62] Liqun Chen and Jiangtao Li. VLR group signatures with indisputable exculpability and efficient revocation. *IJIPSI*, 1(2/3):129–159, 2012.

[63] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[64] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT 2003*, pages 416–432, 2003.

[65] Danfeng Yao and Roberto Tamassia. Compact and anonymous role-based authorization chain. *ACM Trans. Inf. Syst. Sec.*, 12(3):15:1–15:27, 2009.

[66] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[67] Saman Zarandioon, Danfeng (Daphne) Yao, and Vinod Ganapathy. K2C: cryptographic cloud storage with lazy revocation and anonymous access. In *SecureComm 2011*, pages 59–76, 2011.

[68] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *ACM CCS 2004*, pages 354–363, 2004.

[69] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO '04*, pages 41–55, 2004.

[70] Hovav Shacham. *New paradigms in signature schemes*. PhD thesis, Stanford University, 2005.

[71] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathe-*

matics, 156(16):3113 – 3121, 2008. Applications of Algebra to Cryptography.

[72] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(5):1234–1243, 2001.

[73] Xiaoyan Zhu, Haotian Chi, Shunrong Jiang, Xiaosan Lei, and Hui Li. Using dynamic pseudo-IDs to protect privacy in location-based services. In *IEEE ICC '14*, pages 2307–2312, 2014.

[74] Francesco Restuccia, Sajal K. Das, and Jamie Payton. Incentive mechanisms for participatory sensing: Survey and research challenges. *ACM Trans. Sen. Netw.*, 12(2):13:1–13:40, 2016.

[75] L. Cheng, L. Kong, C. Luo, J. Niu, Y. Gu, W. He, and S. Das. False data detection and correction framework for participatory sensing. In *IWQoS '15*, pages 213–218, 2015.

[76] John R. Douceur. The sybil attack. In Peter Druschel, M. Frans Kaashoek, and Antony I. T. Rowstron, editors, *IPTPS '02*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260, 2002.

[77] Ben Lynn. Pbc (pairing-based cryptography) library. https://crypto.stanford.edu/pbc/, 2016. [Online; accessed 16-May-2016].

[78] Angelo De Caro and Vincenzo Iovino. jpbc: Java pairing based cryptography. In *IEEE ISCC '11*, pages 850–855. IEEE, 2011.

[79] Kenji Koyama and Yukio Tsuruoka. Speeding up elliptic cryptosystems by using a signed binary window method. In *CRYPTO '92*, pages 345–357, 1992.

[80] Klaus Potzmader and Johannes Winter *et al.* Group signatures on mobile devices: Practical experiences. In *Trust and Trustworthy Computing*, pages 47–64, 2013.

[81] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000.

Appendix A

8.1 BU-anonymity (Theorem 4.5)

In [16], Boneh and Shacham developed a technique to prove the anonymity by capitalizing the randomness of $(\hat{u}, \hat{v})$ and the ability of backpatching the hash queries (H_g, H_z). Hence, we can employ the core technique that was used in [16] to prove this theorem.

Proof. Suppose algorithm $\mathcal{A}$-breaks the BU-anonymity of SRBE scheme. We build an algorithm $\mathcal{B}$ that breaks the DLIN assumption in $\mathbb{G}_2$. Algorithm $\mathcal{B}$ is given as input a 6-tuple $u, v, h, u^a, v^b, z \in \mathbb{G}_2^6$ where $a, b \xleftarrow{R} \mathbb{Z}_p^*$ and either $z = h^{a+b}$ or z is random in $\mathbb{G}_2$. Now $\mathcal{B}$ interacts with $\mathcal{A}$ according to the BU-Anonymity game (Definition 3.3). Where in setup phase, $\mathcal{B}$ picks two random users $i_0, i_1 \xleftarrow{R} \{1, \cdots, N\}$ and generates the private key gsk_i and revocation token grt_i for user i, where $i \in [1, N] - \{i_0, i_1\}$ according to the `Join` protocol defined in SRBE. For users i_0 and i_1, $\mathcal{B}$ first selects $W \xleftarrow{R} \mathbb{G}_2$, then defines $A_{i_0} = \frac{zW}{h^a}$ and $A_{i_1} = Wh^b$.

During signing query for i_0 or i_1, depending on the user, challenger $\mathcal{B}$ simulates the values of either A_{i_0} or A_{i_1} while generating T_1, T_2 similar as [16] and selects $T_3, T_4 \xleftarrow{R} \mathbb{G}_2$ and $T_5 \xleftarrow{R} \mathbb{G}_1$ to simulate corresponding B_i, C_{ij}, PID_{ij}, where $i \in \{i_0, i_1\}$, $j \in [1, T]$. Then it produces the signature σ by using these values according to the `Sign` procedure. $\mathcal{B}$ also back patches hash queries (H_g, H_z) to ensure consistency. If $\mathcal{A}$ issues any hash queries before back patching, then $\mathcal{B}$ reports failure and aborts. According to [16], σ is a properly distributed signature under signer i's private key.

During challenge phase, $\mathcal{A}$ outputs a message M^*, time period j^* and two users i_0^* and i_1^*, whose are neither corrupted nor revoked at time period j^*. If $\{i_0^*, i_1^*\} \neq \{i_0, i_1\}$, then $\mathcal{B}$ reports failure and aborts. Otherwise, $\mathcal{B}$ picks a random $b \xleftarrow{R} \{0, 1\}$ and generates a signature σ^* using signer i_b's key for M^*, similar as the signing queries in Phase 1. Then $\mathcal{B}$ sends σ^* as the challenge to $\mathcal{A}$.

During output phase, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b. If $b = b'$ then $\mathcal{B}$ outputs 0 (indicating that z is random in G_2); otherwise $\mathcal{B}$ outputs 1 (indicating that $z = h^{a+b}$).

If we assume that, during the simulation of the above interaction framework $\mathcal{B}$ does not abort, we see that $\mathcal{B}$ can break the DLIN assumption in $\mathbb{G}_2$ with an advantage of $\frac{\epsilon}{2}$. We see that, the probability of selecting i_0^* and i_i^* by $\mathcal{A}$ without causing $\mathcal{B}$ to abort is $\frac{1}{n^2}$. Even if $\mathcal{A}$ correctly select i_0^* and i_i^*, the probability of $\mathcal{B}$ to abort due to signing queries is $\frac{q_S q_H}{p}$. So the probability of $\mathcal{B}$ not to abort is $\frac{1}{N^2} - \frac{q_S q_H}{p}$, which makes $\mathcal{B}$ to break the DLIN assumption in $\mathbb{G}_2$ with an advantage of $\frac{\epsilon}{2}(\frac{1}{N^2} - \frac{q_S q_H}{p})$. □

Signatures from different time intervals are unlinkable, hence BU-anonymity is preserved for across time epochs. However, signatures under the same time epoch contain the same pseudoID. Therefore, they are linkable. It remains an open problem to design a group signature scheme with sublinear revocation supporting full anonymity. The challange is that any information that can make revocation tokens pre-computable at the verifier's side needs to be excluded from the SPK, which can be used to break the within-epoch anonymity.

8.2 Traceability (Theorem 4.6)

To prove the traceability theorem, we recall Lemma 1 of group signature scheme with probabilistic revocation [11] as follows.

Lemma 8.1. *Suppose an algorithm $\mathcal{A}$ that is given an instance $(\tilde{g}_1, \tilde{g}_2, \tilde{g}_2^{\gamma}, \cdots, \tilde{g}_2^{\gamma^T})$ and N tuples of $(\tilde{A}_i, x_{i1}, x_{i2}, \cdots, x_{iT})$, $\forall i \in [1, N]$, where $x_{ij} \in \mathbb{Z}_p^*, \forall i \in [1, N], j \in [1, T]$, $\tilde{g}_2 \in \mathbb{G}_2$,*

$\tilde{g}_1 = \psi(\tilde{g}_2)$, $\tilde{A}_i = \tilde{g}_1^{1/[\Pi_{j=1}^{T}(\gamma+x_{ij})]}$, *forges a tuple* $(\tilde{A}_*, \tilde{B}_*, \tilde{C}_*, x_*)$ *for some* $\tilde{A}_* \in \mathbb{G}_1$, $\tilde{B}_* \in \mathbb{G}_2$, $\tilde{C}_* \in \mathbb{G}_2$ *and* $x_* \neq x_{ij}, \forall i \in [1,N], j \in [1,T]$ *such that* $e(\tilde{A}_*, \tilde{B}_*) = e(\tilde{g}_1, \tilde{g}_2)$ *and* $e(\tilde{g}_1, \tilde{B}_*) = e(\tilde{g}_1^{\gamma}\tilde{g}_1^{x_*}, \tilde{C}_*)$, *then there exists an algorithm* $\mathcal{B}$ *solving q-BSDH problem, where* $q = (N+1)T$.

Similar as Boneh-Boyen [61] weak signature scheme, Lemma 8.1 prescribes the security of x_{ij} of GSPR scheme against existential forgery under a weak chosen message attack, when q-BSDH assumption holds. Similar as Boneh-Boyen full signature scheme, we need to show that PID_{ij} of SRBE is secure against existential forgery under a chosen message attack, when q-BSDH assumption holds. Hence, we state the following lemma.

Lemma 8.2. *Suppose an algorithm* $\mathcal{A}$ *that is given an instance* $(\tilde{g}_1, \tilde{g}_2, \tilde{g}_2^{\gamma_1}, \tilde{g}_2^{\gamma_2})$ *and* N *tuples of* $(\tilde{A}_i, PID_{i1}, PID_{i2}, \cdots, PID_{iT})$, $\forall i \in [1,N]$, *where* $PID_{ij} \in \mathbb{Z}_p^*, \forall i \in [1,N], j \in [1,T]$, $\tilde{g}_2 \in \mathbb{G}_2$, $\tilde{g}_1 = \psi(\tilde{g}_2)$, $\tilde{A}_i = \tilde{g}_1^{1/[\Pi_{j=1}^{T}(\gamma_1+\gamma_2\tau_j+PID_{ij})]}$, *forges a tuple* $(\tilde{A}_*, \tilde{B}_*, \tilde{C}_*, PID_*, j_*)$ *for some* $\tilde{A}_* \in \mathbb{G}_1$, $\tilde{B}_* \in \mathbb{G}_2$, $\tilde{C}_* \in \mathbb{G}_2$, $PID_* \in \mathbb{Z}_p^*$ *and* $j_* \in [1,T]$, *such that* $e(\tilde{A}_*, \tilde{B}_*) = e(\tilde{g}_1, \tilde{g}_2)$ *and* $e(\tilde{g}_1, \tilde{B}_*) = e(\tilde{g}_1^{\gamma_1}\tilde{g}_1^{\gamma_2\tau_*}\tilde{g}_1^{PID_*}, \tilde{C}_*)$ *where* $\tau_* = H_z(j_*)$, *then there exists an algorithm* $\mathcal{B}$ *to solve q-BSDH problem, where* $q = (N+1)T$.

Proof. We see that, to forge a tuple, $\mathcal{A}$ can instantiate 2 types of forgers as follows.

Type I Forger. Forges a tuple $(\tilde{A}_*, \tilde{B}_*, \tilde{C}_*, PID_*, j_*)$ for some $\tilde{A}_* \in \mathbb{G}_1$, $\tilde{B}_* \in \mathbb{G}_2$, $\tilde{C}_* \in \mathbb{G}_2$, $\gamma_2\tau_* + PID_* \neq \gamma_2\tau_j + PID_{ij}, \forall i \in [1,N], j \in [1,T]$, such that $e(\tilde{A}_*, \tilde{B}_*) = e(\tilde{g}_1, \tilde{g}_2)$ and $e(\tilde{g}_1, \tilde{B}_*) = e(\tilde{g}_1^{\gamma_1}\tilde{g}_1^{\gamma_2\tau_*}\tilde{g}_1^{PID_*}, \tilde{C}_*)$.

Type II Forger. Forges a tuple $(\tilde{A}_*, \tilde{B}_*, \tilde{C}_*, PID_*, j_*)$ for some $\tilde{A}_* \in \mathbb{G}_1$, $\tilde{B}_* \in \mathbb{G}_2$, $\tilde{C}_* \in \mathbb{G}_2$, $\gamma_2\tau_* + PID_* = \gamma_2\tau_j + PID_{ij}$ but $\tau_* \neq \tau_j, PID_* \neq PID_{ij}$ for some $i \in [1,N], j \in [1,T]$, such that $e(\tilde{A}_*, \tilde{B}_*) = e(\tilde{g}_1, \tilde{g}_2)$ and $e(\tilde{g}_1, \tilde{B}_*) = e(\tilde{g}_1^{\gamma_1}\tilde{g}_1^{\gamma_2\tau_*}\tilde{g}_1^{PID_*}, \tilde{C}_*)$.

Using the similar method used by Boneh and Boyen to prove the security of full signature scheme [61], we can show that, either forger can be used to forge a tuple defined in Lemma 8.1. To give some intuition, one can observe that, Forger I succeeds to forge, only if it finds a PID_*, so that $\gamma_1 = -PID_*$. Forger II succeeds, if it can find some $\tau_* = H_z(j_*), PID_*$ such that, $\tilde{g}_1^{\gamma_2\tau_*}\tilde{g}_1^{PID_*} = \tilde{g}_1^{\gamma_2\tau_j}\tilde{g}_1^{PID_{ij}}$, but $(\tau_*, PID_*) \neq (\tau_j, PID_{ij})$, which implies that, it can extract γ_2 by computing, $\gamma_2 = (PID_{ij} - PID_*)/(\tau_* - \tau_j)$.

Now, if algorithm $\mathcal{B}$ is given with tuple $(\tilde{g}_1, \tilde{g}_2, \tilde{g}_2^{\gamma}, \cdots, \tilde{g}_2^{\gamma^T})$ and N tuples of $(\tilde{A}_i, x_{i1}, x_{i2}, \cdots, x_{iT})$, $\forall i \in [1,N]$, where $x_{ij} \in \mathbb{Z}_p^*, \forall i \in [1,N], j \in [1,T]$, $\tilde{g}_2 \in \mathbb{G}_2$, $\tilde{g}_1 = \psi(\tilde{g}_2)$, $\tilde{A}_i = \tilde{g}_1^{1/[\Pi_{j=1}^{T}(\gamma+x_{ij})]}$, and asked to forge a tuple $(\tilde{A}_*, \tilde{B}_*, \tilde{C}_*, x_*)$ for some $\tilde{A}_* \in \mathbb{G}_1$, $\tilde{B}_* \in \mathbb{G}_2$, $\tilde{C}_* \in \mathbb{G}_2$ and $x_* \neq x_{ij}, \forall i \in [1,N], j \in [1,T]$ such that $e(\tilde{A}_*, \tilde{B}_*) = e(\tilde{g}_1, \tilde{g}_2)$ and $e(\tilde{g}_1, \tilde{B}_*) = e(\tilde{g}_1^{\gamma}\tilde{g}_1^{x_*}, \tilde{C}_*)$, depending on the instantiation of forgers by $\mathcal{A}$, $\mathcal{B}$ can ask $\mathcal{A}$ to forge a tuple, by either defining, $\gamma_1 = \gamma, \gamma_2\tau_j + PID_{ij} = x_{ij}$ or $\gamma_2\tau_j = \gamma, \gamma_1 + PID_{ij} = x_{ij}$. Which contradicts with Lemma 8.1. So, we conclude that Lemma 8.2 holds. □

8.2.1 Proof Theorem 4.6

Proof. The following is an interaction between $\mathcal{A}$ and $\mathcal{B}$.

- **Setup.** Algorithm $\mathcal{B}$ is given two groups $\mathbb{G}_1$, and $\mathbb{G}_2$) with generators g_1, g_2 respectively. $\mathcal{B}$ is also given $w_1 = g_2^{\gamma_1}$, $w_2 = g_2^{\gamma_2}$ and a list of tuples $(SEED_{i1}, SEED_{i2}, A_i, B_i, \{C_{ij}\}), \forall j \in [1,T], \forall i \in [1,N]$. For each signer i, $\mathcal{B}$ sets either $s_i = 0$, means that the given tuple is generated with `Join` protocol (For simplicity, let's assume $\mathcal{B}$ selects $f_i = 1$ in the join protocol for all users), or else $\mathcal{B}$ sets $s_i = 1$ indicating that $(SEED_{i1}, SEED_{i2})$ corresponding to $(A_i, B_i, C_{ij}), \forall j \in [1,T]$ is not known. Then $\mathcal{B}$ runs $\mathcal{A}$, giving it the group public key (g_1, g_2, w_1, w_2) and $(SEED_{i1}, SEED_{i2})$. After that $\mathcal{B}$ answers $\mathcal{A}$'s oracle queries as follows.
- **Queries.** At the beginning of each period j, $\mathcal{A}$ announces the beginning of j to $\mathcal{B}$, so that they both increment j simultaneously. At any time period $j \in [1,T]$, Algorithm $\mathcal{A}$ can make queries to $\mathcal{B}$, as follows.
  - **Signing:** At time period j, Algorithm $\mathcal{A}$ requests a signature on an arbitrary message M for an arbitrary signer i. If $s_i = 0$, then $\mathcal{B}$ computes the signature $\sigma \leftarrow Sign(gpk, gsk_i, M)$ and returns σ to $\mathcal{A}$. If $s_i = 1$, $\mathcal{B}$ selects $(PID_{ij}, \alpha, \beta, \delta)$, computes $(\hat{u}, \hat{v}, T_1, T_2, T_3, T_4, R_1, R_2, R_3, c, s_\alpha, s_\beta, s_\delta)$ and derives a signature $\sigma = (r, PID_{ij}, T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_\delta)$. In addition $\mathcal{B}$, patches the hash oracle. If in case, hash function causes collision, $\mathcal{B}$ declares failure and exits. Otherwise, $\mathcal{B}$ returns σ to $\mathcal{A}$. A signature query can trigger a hash query, which we charge against $\mathcal{A}$'s hash query limit.
  - **Corruption:** Algorithm $\mathcal{A}$ requests the secret key of user i at any time period j. If $s_i = 0$, then $\mathcal{B}$ sets $U \leftarrow U \cup \{i\}$ responds with $(SEED_{i1}, SEED_{i2}, A_i, B_i, C_{ij})$, where $j \in [1,T]$. otherwise $\mathcal{B}$ declares failure and exit.
- **Output.** Finally if algorithm $\mathcal{A}$ is successful, it outputs a forged signature σ^* on a message M^* using tuple

$(A_{i^*}, B_{i^*}, C_{i^*j})$ at any time period j. For the forgery to be non-trivial, i should not be in U. If indeed $\mathcal{B}$ fails to find the signer i^* in U, it outputs σ^*. If $s_i = 1$, $\mathcal{B}$ outputs σ^*, otherwise it declares failure and exits.

As implied by the output phase of the framework above, there are two types of forger algorithm, similar to [16]. Type I forger forges a signature σ on a message M for a user $i \notin [1, N]$. Type II forger forges a signature of user $i \in [1, N]$ whose corruption query is yet to be requested. Hence, similar as [11], against Type I forger, we assign N valid private keys to N and against Type II forger, we randomly choose a signer i' and assign $N-1$ private keys to rest of the $N-1$ signers.

For Type I forgery, if $\mathcal{A}$ succeeds with an advantage of ϵ, $\mathcal{B}$ also succeeds with an advantage of ϵ. But for Type II forgery, $\mathcal{B}$ gains against the BSDH instance only if the signature is signed with the private key of signer i'. Hence, the probability of $\mathcal{B}$ to be succeeded is ϵ/N. For both Type of the forgeries, $\mathcal{B}$ rewinds the interaction framework, between $\mathcal{A}$ and $\mathcal{B}$ to obtain two forged signatures on the same message. According to the forking lemma [16] the probability of $\mathcal{B}$ to be succeeded is at least $(\epsilon' - 1/p)^2/16q_H$, where ϵ' is the probability of successful forgery. After gaining the forged signature $\mathcal{B}$ extracts $(\tilde{A}_*, \tilde{B}_*, \tilde{C}_*, PID_*, j_*)$ (similar as Lemma 5.2 in [16]) and then, using the technique employed in 8.2, $\mathcal{B}$ can compute a new BSDH pair. So $\mathcal{B}$ can break the q-BSDH assumption, by obtaining a new BSDH pair with an advantage of $(\epsilon/N - 1/p)^2/16q_H$. □

8.3 Exculpability (Theorem 4.7)

Proof. If an adversary $\mathcal{A}$ breaks the exculpability game (Definition 3.5) with non-negligible probability, we can construct another polynomial-time algorithm $\mathcal{B}$ to solve DL problem in $\mathbb{G}_2$ with non-negligible probability.

Let us assume that $\mathcal{B}$ is given a DL instance $(\tilde{g}, \tilde{h})$. It then finds $\log_{\tilde{g}\tilde{h}}$ by interacting with $\mathcal{A}$.

Setup. $\mathcal{B}$ performs `KeyGen(1^λ)` as in the scheme, except that she sets $g_2 \longleftarrow \tilde{g}$, $g_1 \longleftarrow \psi(g_2)$. $\mathcal{B}$ stores the group public key gpk, sends gpk, group manager's secret gms, registration list reg to $\mathcal{A}$. It also initializes a list of revocation lists RL_j, where $j \in [1, T]$.

Queries. At the beginning of each period j, $\mathcal{A}$ announces the beginning of j to $\mathcal{B}$, so that they both increment j simultaneously. At any time period $j \in [1, T]$, Algorithm $\mathcal{A}$ issues the following queries to $\mathcal{B}$.

- **Join:** When $\mathcal{A}$ requests for creating a new group member, $\mathcal{B}$ performs **Join** protocol as the new member with $\mathcal{A}$, except that it sets $F_{i^*} \longleftarrow \psi(\tilde{h})$ for a random user i^*. $\mathcal{B}$ also simulates the proof of knowledge of $\log_{g_1} F_{i^*}$. So the signer's secret during join protocol, $f_i^* = \log_{g_1} F_{i^*} = \log_{\tilde{g}} \tilde{h}$, but $\mathcal{A}$ does not know its value.
- **Hash queries:** At any time, $\mathcal{A}$ can query the hash functions H_z. Algorithm $\mathcal{B}$ responds with random values while ensuring consistency.
- **Signing:** If $i \neq i^*$, $\mathcal{B}$ returns the signature signed as in the scheme. Otherwise, $\mathcal{B}$ picks $\alpha, \beta', \delta' \longleftarrow \mathbb{Z}_p^*$ and makes the following assignments:

$$T_1 = u^{\alpha}, \quad T_2 = A_i v^{\alpha},$$
$$T_3 = B_i'^{\beta'}, \quad T_4 = C_{ij}'^{\delta'}.$$

  Let $\beta = \beta'/f_i$, $\delta = \delta'/f_i$. Then we observe that, $T_1 = u^{\alpha}, T_2 = A_i v^{\alpha}, T_3 = B_i^{\beta}, T_4 = C_{ij}^{\delta}$.

  Algorithm $\mathcal{B}$ then selects $r_\alpha, r_\beta, r_\delta \xleftarrow{R} \mathbb{Z}_p^*$ and computes the corresponding R_1, R_2, R_3. In the unlikely event $\mathcal{A}$ has already issued a hash query for $H_z(gpk, M, j, PID_{ij}T_1, T_2, T_3, T_4, R_1, R_2, R_3)$, then $\mathcal{B}$ reports failure and terminates. Otherwise $\mathcal{B}$ defines,

$$H_z(gpk, M, j, PID_{ij}, T_1, T_2, T_3, T_4, R_1, R_2, R_3) = c$$

  Algorithm $\mathcal{B}$ then computes the signature σ as $\sigma = (r, PID_{ij}, T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_\delta)$ and gives σ to $\mathcal{A}$. According to [16], σ is a properly distributed signature under signer i's private key.
- **Corruption:** $\mathcal{B}$ returns the secret secret key gsk_i to $\mathcal{A}$ and updates the current and future revocation lists (RL_k, $\forall k \in [j, T]$) with corresponding revocation handles at time period k.

Forge. Algorithm $\mathcal{A}$ outputs a message M^*, time period j^*, a signature σ^* and a signer i^*.

$\mathcal{B}$ has an advantage against the given DL instance if T_5 corresponding to σ^*, indeed represents i^*. As i^* looks random for $\mathcal{A}$, so the probability that i^* is chosen from $[1, N]$ is at least $1/N$.

If we assume that $\mathcal{A}$ wins the exculpability game, we can state that $T_3 = B_i'^{\beta'/f_i^*}$. Since this statement is indisputable, by employing forking lemma [81], after a polynomial reply of algorithm $\mathcal{A}$, $\mathcal{B}$ can extract f_{i^*}. Consequently, it finds $\log_{\tilde{g}} \tilde{h}$, the solution for her DL instance, with non-negligible probability in polynomial time. □