Group Signatures with Probabilistic Revocation: A Computationally-Scalable Approach for Providing Privacy-Preserving Authentication

Vireshwar Kumar[1], He Li[1], Jung-Min (Jerry) Park[1], Kaigui Bian[2], Yaling Yang[1]

[1] Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA, USA
{viresh, heli, jungmin, yyang8}@vt.edu

[2] Electronics Engineering and Computer Science
Peking University
Beijing, China
bkg@pku.edu.cn

ABSTRACT

Group signatures (GSs) is an elegant approach for providing privacy-preserving authentication. Unfortunately, modern GS schemes have limited practical value for use in large networks due to the high computational complexity of their revocation check procedures. We propose a novel GS scheme called the *Group Signatures with Probabilistic Revocation* (GSPR), which significantly improves scalability with regard to revocation. GSPR employs the novel notion of *probabilistic revocation*, which enables the verifier to check the revocation status of the private key of a given signature very efficiently. However, GSPR's revocation check procedure produces probabilistic results, which may include false positive results but *no* false negative results. GSPR includes a procedure that can be used to iteratively decrease the probability of false positives. GSPR makes an advantageous tradeoff between computational complexity and communication overhead, resulting in a GS scheme that offers a number of practical advantages over the prior art. We provide a proof of security for GSPR in the random oracle model using the decisional linear assumption and the bilinear strong Diffie-Hellman assumption.

Categories and Subject Descriptors
E.3 [**Data**]: Data Encryption—*Public Key Cryptosystems*; K.4.1 [**Computer and Society**]: Public Policy Issues—*Privacy*

Keywords
Privacy-preserving authentication; group signature; probabilistic revocation.

1. INTRODUCTION

The notion of authentication is to enable a sender to prove her identity to a distant communication partner and/or to show that she is the origin of the transmitted data. This security attribute is essential to most of today's applications that rely on digital communications over insecure networks. In some applications, however, authentication is not sufficient, and in addition to authentication, the sender's privacy need to be protected—the combination of these two attributes is often referred to as *privacy-preserving authentication* (PPA). PPA schemes are needed in applications where the verifiers should not learn the actual identity of the sender (i.e., signer), and are willing to accept an authentication artifact (i.e., signature) that is verifiably linked to an anonymous sender, knowing that the sender's identity can be revealed by a trusted third party, if disputes need to be resolved. A wide variety of applications require PPA, including safety applications for vehicular networks, identity escrow schemes, anonymous credential systems, remote attestation of computing platforms, and device-to-device communications in the Internet-of-Things (IoT) paradigm.

For deployment in large networks, PPA protocols need to rely on public-key cryptography. In public-key cryptosystem-based PPA protocols, there are three entities that interact with each other: *signer*, *verifier*, and *group manager*. The roles of the signer and the verifier are obvious. The group manager plays an important role. During the initialization process, the group manager generates parameters, and certificates (e.g., public-key certificates) and the private signing key of each group member. Most importantly, the group manager has the ability to reveal the signer's true identity if a dispute needs to be resolved. PPA schemes can be broadly categorized into two approaches: *pseudonym-based signatures* (PSs) [20] and *group signatures* (GSs) [6].

In PSs, legacy public-key cryptosystems (e.g., RSA) are used. The group manager provides the signer with a list of pseudonyms and the corresponding private keys, public keys, and public-key certificates. The signer creates a signature based on her pseudonym, and replaces her pseudonym with a new one periodically to preserve anonymity. Although the PS approach is straightforward, it has a number of drawbacks. Because each pseudonym needs to be used with its unique set of private and public keys and a certificate, key management and distribution become a very onerous burden in large networks [12].

GSs do not require public-key certificates, and hence do not need a certificate distribution framework. In GS, each signer is a member of a group, and she is provided with a private key tuple by the group manager. Using this tuple, the

CCS'15, October 12–16, 2015, Denver, Colorado, USA.

DOI: http://dx.doi.org/10.1145/2810103.2813602.

signer generates signatures without revealing her true identity to the verifier. In the case of a conflict, the signature can be "opened" by the group manager, and the identity of the signer is revealed. The most practical GS schemes support verifier-local revocation (VLR) [3, 6, 16]. To perform VLR, the group manager generates a revocation token for each signer (which is a portion of the private key), publishes it in a revocation list, and distributes the revocation list to the verifiers. To check the revocation status of the private key used to generate the received signature, the verifier performs the revocation check procedure. This procedure involves going through the revocation list, and checking whether any of the revocation tokens contained therein can be mapped to the received signature. This means that the computation time for the revocation check procedure increases linearly with the number of revoked private keys. Moreover, the computational cost of the procedure for each revocation token is expensive. After a thorough and comprehensive analysis of existing GSs, the authors of [13] recently concluded that *revocation remains the major performance bottleneck of modern GS schemes*, and that further research is urgently needed to design schemes offering better scalability with regard to revocation.

In this paper, we propose a novel VLR GS scheme called *Group Signatures with Probabilistic Revocation* (GSPR). As its name implies, the most striking attribute of GSPR is that it supports *probabilistic revocation*. That is, GSPR's revocation check procedure does not produce deterministic results, but instead produces probabilistic results, which may include false positive (i.e., false alarm) results but *no* false negative results. Here, a false negative result refers to an instance in which the revocation check algorithm fails to detect that the revocation token associated with the received signature is included in the revocation list. GSPR includes a procedure that can be used to iteratively decrease the probability of false alarms. The use of probabilistic revocation (instead of deterministic revocation) enables GSPR to elegantly address the primary performance bottleneck of GSs—i.e., enable very efficient revocation checking with only a modest increase in the signature size. In fact, GSPR's revocation check time does not grow linearly with the number of revoked keys.

The dramatic improvement in the computational efficiency of the revocation check procedure is made possible by the use of "alias codes". Each alias code is a vector of $+1$s and -1s with desirable cross-correlation properties, and each alias code is mapped to an "alias token" (which is equivalent to a revocation token in legacy VLR GS schemes) included in each signature. The group manager creates a "revocation code" (which is equivalent to a revocation list) by adding all of the alias codes mapped to revoked alias tokens, and then distributes this to the verifiers. The verifier performs the revocation check procedure by first mapping the signature's alias token to an alias code, and then computing the cross correlation of the alias code and the revocation code. Note that the verifier is able to check whether a particular alias code is included in the revocation code in a *single* cross-correlation operation, and thus avoids the burden of legacy GS schemes in which the verifier needs to iteratively check each revocation token in the revocation list. Because of the probabilistic nature of the revocation check procedure, its result is not guaranteed to be correct with certainty, but only with a certain probability.

The paper's main contributions are summarized below.

- We propose a novel VLR GS scheme called *Group Signatures with Probabilistic Revocation* which significantly reduces the computational complexity of the revocation check procedure compared to the prior art.
- We propose the novel concept of probabilistic revocation which makes an advantageous tradeoff between computational complexity and communication overhead. This tradeoff enables GSPR to have significantly better scalability in terms of revocation compared to the prior art.
- We provide a comprehensive security analysis of GSPR in the random oracle model using standard complexity assumptions often used in evaluating legacy schemes.

The rest of this paper is organized as follows. We provide the security assumptions in Section 2, and present the model and security definitions in Section 3. We present the details of GSPR in Section 4, and analyze its security properties in Section 5. We perform the computational and communication overhead analysis of GSPR in Section 6. We discuss GSPR in the context of safety applications for vehicular networks in Section 7. We discuss the related work in Section 8, and conclude the paper in Section 9.

2. PRELIMINARIES

The proposed scheme is constructed in cyclic groups with a computable bilinear map [5]. Moreover, the security of the proposed scheme is proved in the random oracle model using the Decisional Linear (DLIN) assumption [4] and the q-Bilinear Strong Diffie-Hellman (BSDH) assumption [10]. In this section, we review the definitions of bilinear groups and of the complexity assumptions.

Definition 1. **(Bilinear Groups)**: $(\mathbb{G}_1, \mathbb{G}_2)$ is called a bilinear group pair, if there exists a group $\mathbb{G}_T$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the following properties:

1. $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p;
2. g_1 is a generator of $\mathbb{G}_1$, and g_2 is a generator of $\mathbb{G}_2$;
3. ψ is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$;
4. e is an efficiently computable bilinear map with the following properties:
   - Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$, $\forall u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$, where $\mathbb{Z}_p^*$ represents the set of integers modulo p; and
   - Non-degenerate: $e(g_1, g_2) \neq 1$.

When $\mathbb{G}_1 = \mathbb{G}_2$, ψ is an identity map. On the other hand, when $\mathbb{G}_1 \neq \mathbb{G}_2$, certain families of non-supersingular elliptic curves can be used for efficient implementation of bilinear groups, and ψ can be implemented by a trace map [14].

Definition 2. **(DLIN Assumption)**: Given $u_0, u_1, h, u_0^a, u_1^b, Z \in \mathbb{G}_2$, where $a, b \in \mathbb{Z}_p^*$, as input for each probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the probability with which $\mathcal{A}$ is able to differentiate whether $Z = h^{a+b}$ or $Z \xleftarrow{R} \mathbb{G}_2$ is negligibly small. Here, $\xleftarrow{R}$ represents a random selection.

Definition 3. **(BSDH Assumption)**: Given a $(q+2)$-tuple $(g_1, g_2, g_2^{\gamma}, \cdots, g_2^{\gamma^q})$ as input for each PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ outputs a pair $(e(g_1, g_2)^{1/(\gamma+x)}, x)$, where $x \in \mathbb{Z}_p^*$, $g_2 \xleftarrow{R} \mathbb{G}_2$, $g_1 = \psi(g_2)$, and $\gamma \xleftarrow{R} \mathbb{Z}_p^*$, is negligibly small.

3. MODEL AND SECURITY DEFINITIONS

In this section, we describe the algorithms that make up GSPR, and review the security properties of GSs.

Definition 4. **(Group Signatures with Probabilistic Revocation)**: It is composed of the following algorithms.

- $\mathsf{KeyGen}(\lambda)$: With the security parameter, $\lambda \in \mathbb{N}$, this algorithm generates a group public key gpk, and a group manager's secret gms. Here, $\mathbb{N}$ represents the set of natural numbers.
- $\mathsf{Join}(gms, i, m)$: To add the signer $i \in [1, n]$, where n is the total number of signers in the network, as a member of the group with the secret gms, this algorithm generates a set of m alias tokens, x_{ik}, $\forall k \in [1, m]$, a corresponding secret/private key $\boldsymbol{gsk}_i$ and a corresponding revocation token $\boldsymbol{grt}_i$, and makes an entry into a registration list $\boldsymbol{reg}_i$. In this paper, we use the terms "secret key" and "private key" interchangeably.
- $\mathsf{Sign}(gpk, \boldsymbol{gsk}_i, M)$: With the group public key gpk, and the signer's secret key $\boldsymbol{gsk}_i$, this algorithm generates signature σ with alias token x_{ik} on message M.
- $\mathsf{Verify}(gpk, RC, \sigma, M)$: If both of the following sub-algorithms produce an output value of `valid`, this algorithm outputs the value `valid`; otherwise, it outputs the value invalid.
  - $\mathsf{SignCheck}(gpk, \sigma, M)$: With the group public key gpk and a purported signature σ on a message M, this sub-algorithm outputs the value `valid` if σ is an honest signature on M; otherwise, it outputs the value invalid.
  - $\mathsf{RevCheck}(RC, \sigma)$: With a revocation code RC and a purported signature σ, this sub-algorithm outputs the value `valid` if the alias token x_{ik} embedded in σ is determined to be unrevoked; otherwise, it outputs the value invalid.
- $\mathsf{Revoke}(\boldsymbol{grt}_i, RC)$: This algorithm updates the revocation code RC using the revocation token $\boldsymbol{grt}_i$ if the membership of signer i is to be revoked. Here, revoking the membership of the signer is equivalent to revoking her private key and revoking all of her alias tokens.
- $\mathsf{Open}(\boldsymbol{reg}, \sigma, M)$: Given a valid signature σ on a message M, created by a signer $i \in [1, n]$, this algorithm outputs the signer's identity i.

In this paper, we assume that the group manager runs KeyGen to set-up the group, Join to add a signer to the group, Revoke to revoke a private key of a signer, and Open to open a signature. The signer runs Sign to sign a message, and the verifier runs Verify to verify a signed message.

In the following discussion, we review the three attributes of GSs as per the definitions given in [2].

- *Correctness*: This ensures the following properties.
  - *Signature Correctness*: This ensures that if a signature is generated by an honest signer, the signature check algorithm (i.e., SignCheck) outputs the value `valid`.
  - *Identity Correctness*: This ensures that if a signature is generated by an honest signer, the group manager correctly reveals the identity of the signer using the signature open algorithm (i.e., Open).
  - *Revocation Correctness*: This ensures that if a signature is generated by an honest signer using an unrevoked private key, the revocation check algorithm (i.e., RevCheck) outputs the value `valid`.
- *Anonymity*: This property ensures that no party except the group manager is able to identify the signer of a given signature.
- *Traceability*: This property requires that no colluding set of signers (even consisting of the entire group) can create signatures that cannot be traced back to a signer in the group, or signatures that cannot be traced back to some member of the colluding set.

The revocation correctness property is not considered a core security property in most GSs. However, it is an important property to consider in evaluating our proposed scheme, GSPR, with respect to other GS schemes. GSPR satisfies all of the security properties listed above with the exception of the revocation correctness property. One of the intrinsic attributes of GSPR that distinguishes it from all other GSs is that it satisfies the revocation correctness property with a certain probability, but not with certainty. GSPR exploits the computational efficiency advantage of probabilistic algorithm to significantly reduce the computation cost of the revocation check procedure. Below, we provide formal definitions of the security properties mentioned above.

Definition 5. **(Signature Correctness)**: It requires that for all $\lambda, n \in \mathbb{N}$, all (gpk, gms) obtained by KeyGen, all $(\boldsymbol{gsk}_i, \boldsymbol{grt}_i, \boldsymbol{reg}_i)$ obtained by Join for any $i \in [1, n]$, and all $M \in \{0,1\}^*$,

$$\mathsf{SignCheck}(gpk, \mathsf{Sign}(gpk, \boldsymbol{gsk}_i, M), M) = \texttt{valid}.$$

Definition 6. **(Identity Correctness)**: It requires that for all $\lambda, n \in \mathbb{N}$, all (gpk, gms) obtained by KeyGen, all $(\boldsymbol{gsk}_i, \boldsymbol{grt}_i, \boldsymbol{reg}_i)$ obtained by Join for any $i \in [1, n]$, and all $M \in \{0,1\}^*$,

$$\mathsf{Open}(\boldsymbol{reg}, \mathsf{Sign}(gpk, \boldsymbol{gsk}_i, M), M) = i.$$

Definition 7. **(Revocation Correctness)**: It requires that for all $\lambda, n \in \mathbb{N}$, all (gpk, gms) obtained by KeyGen, all $(\boldsymbol{gsk}_i, \boldsymbol{grt}_i, \boldsymbol{reg}_i)$ obtained by Join for any $i \in [1, n]$, and all $M \in \{0,1\}^*$,

$$\mathsf{RevCheck}(RC, \mathsf{Sign}(gpk, \boldsymbol{gsk}_i, M)) = \texttt{valid},$$

implies that the private key of the signer i is not revoked.

Definition 8. **(Anonymity)**: It requires that for each PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ on winning the following game is negligibly small.

- *Setup*: The challenger runs $\mathsf{KeyGen}(\lambda)$ and $\mathsf{Join}(gms, i, m)$, $\forall i \in [1, n]$. He obtains gpk, $\boldsymbol{gsk}$, and $\boldsymbol{reg}$. He provides the algorithm $\mathcal{A}$ with gpk.
- *Queries-Phase I*: $\mathcal{A}$ queries the challenger about the following.
  - *Signing*: $\mathcal{A}$ requests a signature on an arbitrary message M for an arbitrary member i. The challenger responds with the corresponding signature.
  - *Corruption*: $\mathcal{A}$ requests the secret key of an arbitrary member i. The challenger responds with the key $\boldsymbol{gsk}_i$.
  - *Opening*: $\mathcal{A}$ requests the identity of the signer by providing a message M and its valid signature σ created by signer $i \in [1, n]$. The challenger responds with the signer's identity i.
- *Challenge*: $\mathcal{A}$ outputs a message M and two members i_0 and i_1 with the restriction that the corruption of i_0 and i_1 have not been requested. The challenger chooses $\phi \xleftarrow{R} \{0,1\}$, and responds with the signature σ^* on M^* of member i_ϕ.

- *Queries-Phase II (Restricted Queries)*: After obtaining the challenge, $\mathcal{A}$ can make additional queries of signing, corruption and opening, except the corruption queries of i_0 and i_1, and the opening query of the signature σ^* on M^*.
- *Output*: $\mathcal{A}$ outputs a bit ϕ' indicating its guess of ϕ.

$\mathcal{A}$ wins the anonymity game if $\phi' = \phi$. The advantage of $\mathcal{A}$ is defined as $|\Pr(\phi' = \phi) - 1/2|$.

Definition 9. **(Traceability)**: It requires that for each PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligibly small.

- *Setup*: The challenger runs $\mathsf{KeyGen}(\lambda)$ and $\mathsf{Join}(gms, i, m)$, $\forall i \in [1, n]$. He obtains gpk $\boldsymbol{gsk}$, and $\boldsymbol{reg}$. He provides $\mathcal{A}$ with gpk, and sets U as empty.
- *Queries*: $\mathcal{A}$ queries the challenger about the following.
  - *Signing*: $\mathcal{A}$ requests a signature on an arbitrary message M for an arbitrary member i. The challenger responds with the corresponding signature.
  - *Corruption*: $\mathcal{A}$ requests the secret key of an arbitrary member i. The challenger adds i to U, and responds with the key $\boldsymbol{gsk}_i$.
- *Output*: $\mathcal{A}$ outputs a message M^* and a signature σ^*.

$\mathcal{A}$ wins the game if:

1. $\mathsf{SignCheck}(gpk, \sigma^*, M^*) = \texttt{valid}$;
2. σ^* is traced to a member outside of U or the trace is failure; and
3. $\mathcal{A}$ did not obtain σ^* by making a signing query on M^*.

4. PROPOSED SCHEME: GSPR

Before describing the low-level details of GSPR, we discuss the motivation behind the concept of probabilistic revocation, which is one of the distinguishing attributes of GSPR.

4.1 Motivation for Probabilistic Revocation

In the GSs supporting VLR [3, 6, 16], the group manager includes a revocation token corresponding to each revoked private key in a revocation list, and distributes the revocation list to the verifier. In each VLR based GS scheme, there is an associated implicit tracing algorithm which utilizes the revocation token to link a signature to a revoked private key using which the signature is generated. This implicit algorithm requires several exponentiation and/or bilinear map operations which are computationally expensive. In the revocation check procedure, the verifier performs this implicit tracing algorithm between the received signature, and each revocation token in the revocation list. This means that the computation time for the revocation check procedure of a signature increases linearly with the number of revoked private keys. Hence, the revocation check procedure becomes the major bottleneck in the application of VLR based GSs in real systems with large number of signers along with possibility of large number revoked private keys.

In this paper, we propose a VLR based GS, called *Group Signatures with Probabilistic Revocation* (GSPR), in which an alias token is embedded into the group signature generated by a signer in such a way that it can be utilized for the purpose of revocation check procedure. GSPR significantly reduces the computation complexity of the revocation check procedure by adopting two techniques. Firstly, it reduces the computation cost of executing the implicit tracing algorithm by using the alias tokens in generating signatures. Secondly, it enables the verifier to check the revocation status of an alias token in a single step, instead of requiring the verifier to sequentially go through the revocation list and execute the implicit tracing algorithm for each revocation token included in the revocation list.

Specifically, the group manager issues a set of alias tokens corresponding to a private key of the signer, and the signer embeds an alias token in each of its generated signatures. When the private key of the signer is revoked, all its corresponding alias tokens are added to a revocation list. Further, each alias token is mapped to an "alias code". The group manager performs sample-by-sample addition of all the alias codes corresponding to the alias tokens in the revocation list to generate one code called the "revocation code". The revocation code, instead of the revocation list, is provided to the verifier. When the verifier receives a particular signature with a particular alias token, he generates the alias code corresponding to the alias token. The verifier computes the cross correlation of the alias code and the revocation code. If the value of correlation exceeds a particular threshold, the verifier presumes that the alias code (of the signature being verified) is used to generate the revocation code, and in turn concludes that the signature is invalid because it is associated with a revoked alias token. Otherwise, the verifier concludes that the signature is valid.

The motivation for using alias codes comes from direct-sequence spread spectrum (DSSS) systems used in communications [17]. DSSS is a modulation technique that enables the receiver to remove undue interference and recover the correct information from an aggregate of multiple signals even when multiple transmitters send information simultaneously over a single channel. Information recovery is made possible with the use of specially-crafted spreading codes. Further, we discuss the specific technical details of GSPR.

4.2 Technical Details

For a given security parameter $\lambda \in \mathbb{N}$, we consider a bilinear group pair ($\mathbb{G}_1$,$\mathbb{G}_2$) with isomorphism ψ. We represent $H_z : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H_g : \{0,1\}^* \rightarrow \mathbb{G}_2^2$ as collision resistant hash functions treated as random oracles. Also, we consider a set of alias codes, $\mathbb{C}_p$. The order of $\mathbb{C}_p$ is p which is equal to the order of $\mathbb{Z}_p^*$. Each element in $\mathbb{C}_p$ is an alias code which is a vector of $+1$s and -1s of length l. Further, we define a mapping function $F_c : \mathbb{Z}_p^* \rightarrow \mathbb{C}_p$ using which an alias token in $\mathbb{Z}_p^*$ can be mapped to an alias code in $\mathbb{C}_p$. The details of $\mathbb{C}_p$ and F_c are discussed in Section 5.4.1. $\mathbb{G}_1, \mathbb{G}_2, \psi, H_z, H_g, \mathbb{C}_p$ and F_c are considered public knowledge. In the following paragraphs, we define the algorithms that make up GSPR.

$\mathsf{KeyGen}(\lambda)$: With the security parameter $\lambda \in \mathbb{N}$, this algorithm 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 $\gamma \xleftarrow{R} \mathbb{Z}_p^*$, and compute $w_k = g_2^{\gamma^k}$, $\forall k \in [0, m]$. Note that $w_0 = g_2$.

The group public key is $gpk = (g_1, g_2, w_1, w_2, \cdots w_m)$. The secret belonging only to the group manager is given by $gms = \gamma$. The output of this algorithm is (gpk, gms).

$\mathsf{Join}(gms, i, m)$: This algorithm adds the signer i as a member of the group with the group manager's secret gms, and generates m alias tokens for signer i, and a corresponding secret key $\boldsymbol{gsk}_i$. This algorithm also generates a revocation

token $\boldsymbol{grt}_i$ for signer i, and an entry in the registration list $\boldsymbol{reg}_i$ using the following steps.

1. Select $y_i \xleftarrow{R} \mathbb{Z}_p^*$.
2. Compute the set of m alias tokens,
$$\boldsymbol{X}_i = \{x_{ik} : x_{ik} = H_z(y_i, k), \forall k \in [1, m]\}, \quad (1)$$
where k^{th} alias token of signer i is represented by x_{ik}.
3. Compute $\pi_i = \prod_{k=1}^{m}(\gamma + x_{ik})$, and calculate
$$A_i = g_1^{1/\pi_i}. \quad (2)$$
In the unlikely case, if $\pi_i = 0$, restart from Step 1.

For signer i, the secret key is $\boldsymbol{gsk}_i = (A_i, y_i)$, the revocation token is $\boldsymbol{grt}_i = \boldsymbol{X}_i$, and the entry in the registration list is $\boldsymbol{reg}_i = \boldsymbol{X}_i$. Note that only the group manager has access to $\boldsymbol{reg}$. The output of this algorithm is $(\boldsymbol{gsk}_i, \boldsymbol{grt}_i, \boldsymbol{reg}_i)$.

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

1. Generate the following parameters.
   (a) Compute the alias tokens $\boldsymbol{X}_i$ using equation (1).
   (b) Define $\pi_i = \prod_{k=1}^{m}(\gamma + x_{ik}) = \sum_{k=0}^{m} a_k \gamma^k$, where $a_0, a_1, \cdots, a_m \in \mathbb{Z}_p^*$ are the coefficients of the polynomial π_i with the variable γ, and compute
   $$B_i = g_2^{\pi_i} = \prod_{k=0}^{m} w_k^{a_k}. \quad (3)$$
   (c) For each $x_{ik} \in \boldsymbol{X}_i$, define $\pi_i/(\gamma + x_{ik}) = \prod_{j=1, j\neq k}^{m} (\gamma + x_{ij}) = \sum_{j=0}^{m-1} b_j \gamma^j$, where $b_0, b_1, \cdots, b_{m-1} \in \mathbb{Z}_p^*$ are the coefficients, and compute
   $$C_{ik} = g_2^{\pi_i/(\gamma + x_{ik})} = \prod_{j=0}^{m-1} w_j^{b_j}. \quad (4)$$
2. Select a tuple $(A_i, B_i, C_{ik}, x_{ik})$ by selecting some value of $k \in [1, m]$. The signer utilizes a particular k to sign all its signatures during some time interval. After this time interval, she discards the alias token. When the signer exhausts all its alias tokens, she runs the Join algorithm again to fetch new secret key, and computes corresponding set of new alias tokens.
3. Compute $(\hat{u}, \hat{v}) = H_g(gpk, M, x_{ik})$, and calculate their images in $\mathbb{G}_1$, such that $u = \psi(\hat{u})$ and $v = \psi(\hat{v})$.
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 $T_4 = C_{ik}^\delta$.
5. Compute the signature of knowledge (SPK) [16] which is expressed as follows.
$$\begin{aligned} V = SPK\{&(\alpha, \beta, \delta, x_{ik}, A_i, B_i, C_{ik}) : \\ &T_1 = u^\alpha, T_2 = A_i v^\alpha, T_3 = B_i^\beta, T_4 = C_{ik}^\delta, \\ &e(A_i, B_i) = e(g_1, g_2), e(g_1, B_i) = e(g_1^\gamma g_1^{x_{ik}}, C_{ik})\}(M) \\ = SPK\{&(\alpha, \beta, \delta, x_{ik}, A_i, B_i, C_{ik}) : \\ &T_1 = u^\alpha, e(T_2, T_3) = e(v, T_3)^\alpha \ e(g_1, g_2)^\beta, \\ &1 = e(g_1, T_3)^\delta \ e(\psi(w_1) g_1^{x_{ik}}, T_4)^{-\beta}\}(M). \end{aligned} \quad (5)$$
This SPK is computed with the Fiat-Shamir heuristic method [9] using the following steps.
   (a) Select binding factors $r_\alpha, r_\beta, r_\delta \xleftarrow{R} \mathbb{Z}_p^*$, and compute
   $$\begin{aligned} R_1 &= u^{r_\alpha}, R_2 = e(v, T_3)^{r_\alpha} \ e(g_1, g_2)^{r_\beta}, \\ R_3 &= e(g_1, T_3)^{r_\delta} \ e(\psi(w_1) g_1^{x_{ik}}, T_4)^{-r_\beta}. \end{aligned} \quad (6)$$
   (b) Compute the challenge c as
   $$c = H_z(gpk, M, x_{ik}, T_1, T_2, T_3, T_4, R_1, R_2, R_3).$$
   (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
$$\sigma = (x_{ik}, T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_\delta). \quad (7)$$

Verify(gpk, RC, σ, M): The verification algorithm takes as input the group public key gpk, the revocation code RC, the signature σ, and the message M. Using the following sub-algorithms, it verifies two things: (1) whether the signature was honestly generated, and (2) revocation status of the alias token used to generate the signature. If both the sub-algorithms output `valid`, this algorithm outputs `valid`; otherwise it outputs invalid.

- **SignCheck**(gpk, σ, M): With the group public key gpk and a purported signature σ on a message M, this sub-algorithm outputs `valid` if σ is an honest signature on M. This is checked using the following steps.
  1. Compute $(\hat{u}, \hat{v}) = H_g(gpk, M, x_{ik})$, and calculate their images in $\mathbb{G}_1$, i.e., $u = \psi(\hat{u})$ and $v = \psi(\hat{v})$.
  2. Retrieve:
  $$\begin{aligned} &\widetilde{R}_1 = u^{s_\alpha} T_1^{-c}, \widetilde{R}_2 = e(v, T_3)^{s_\alpha} e(g_1, g_2)^{s_\beta} e(T_2, T_3)^{-c} \\ &\widetilde{R}_3 = e(g_1, T_3)^{s_\delta} \ e(\psi(w_1) g_1^{x_{ik}}, T_4)^{-s_\beta}. \end{aligned} \quad (8)$$
  3. Check the correctness of the challenge c as
  $$c \stackrel{?}{=} H_z(gpk, M, x_{ik}, T_1, T_2, T_3, T_4, \widetilde{R}_1, \widetilde{R}_2, \widetilde{R}_3).$$
  If the above equation holds, this sub-algorithm outputs `valid`; otherwise, it outputs invalid.
- **RevCheck**(RC, σ): The inputs to the revocation check algorithm are the alias token x_{ik} embedded in the signature σ, and the revocation code, RC. The purpose of this sub-algorithm is to check whether the alias token, x_{ik}, has been revoked or not, which is accomplished using the following steps.
  1. Map x_{ik} to the corresponding alias code s_{ik}, i.e., compute $s_{ik} = F_c(x_{ik})$, where s_{ik} is a column vector of length l of samples of $+1$s and -1s.
  2. Compute the value of the decision variable, $z = \frac{1}{l} s_{ik}^T RC$, where s_{ik}^T is the transpose of s_{ik}.
  3. Output invalid if $z \geq \tau$, where τ is a pre-determined threshold; otherwise, output `valid`.

Revoke$(\boldsymbol{grt}_i, RC)$: The inputs to this algorithm are the revocation token of the signer, $\boldsymbol{grt}_i$, and the current revocation code, RC. To revoke signer i, the group manager updates the revocation code using the following steps.

1. Map each $x_{ik} \in \boldsymbol{grt}_i$ to the corresponding alias code s_{ik}, i.e., compute $s_{ik} = F_c(x_{ik})$ for $k = 1, 2 \cdots m$.
2. Compute the code, $\bar{s}_i$, by adding all the unique alias codes corresponding to the revoked alias tokens such that $\bar{s}_i = \sum_{k=1}^{m} s_{ik}$.
3. Update the revocation code as $RC = RC + \bar{s}_i$.

Open$(\boldsymbol{reg}, \sigma, M)$: With the valid signature σ on message M, the actual signer of the signature is identified using the following step.

1. Search the registration list $\boldsymbol{reg}$ to find signer i that has generated signature σ with the alias token x_{ik}.
2. If a match is successfully found, output i; otherwise, output 0 to indicate a failure.

5. SECURITY ANALYSIS

5.1 Signature and Identity Correctness

It can be shown that GSPR satisfies the signature correctness and the identity correctness properties. Security proofs for these properties can be constructed using the frameworks discussed in [6]. We omit the proofs for the sake of brevity of the paper.

5.2 Anonymity

THEOREM 1. *In the random oracle model, suppose an algorithm $\mathcal{A}$ breaks the anonymity of GSPR with advantage ϵ after q_H hash queries and q_S signing queries, then there exists an algorithm $\mathcal{B}$ that breaks the DLIN assumption with the advantage $(1/n^2 - q_S q_H/p)\epsilon/2$.*

This theorem prescribes that GSPR satisfies the anonymity property in the random oracle model when the DLIN assumption is presumed. In [6], the core technique used in the proof of anonymity is the randomness of $(\hat{u}, \hat{v})$ such that the challenger can backpatch the hash oracle. GSPR also preserves the randomness of $(\hat{u}, \hat{v})$. Hence, we can employ the same technique, and the proof construction method used in [6] to prove Theorem 1. However, here we omit the proof due to space constraints.

Note that within a time interval, the signer uses the same alias token to generate all the signatures, and hence those signatures can be linked to the same signer. However, the signer utilizes different alias tokens in different time intervals, and thus non-linkability is preserved between different time intervals. For many applications, the duration of each time interval is small (e.g., 1 minute in vehicular networks [20]), resulting in only a few linkable signatures.

In GSPR, all of the previous signatures generated using a revoked private key can be linked together using the implicit tracing algorithm. The scheme proposed in [6] as well as most other VLR schemes share this drawback. This drawback can be mitigated in a number of ways, including the use of time-stamped parameters [16] or the use of accumulators [21]. However, these methods incur additional overhead that may be unacceptable in many applications.

5.3 Traceability

We consider traceability property of GSPR in Theorem 2, and utilize Lemma 1 to prove it.

LEMMA 1. *Suppose an algorithm $\mathcal{A}$ which is given an instance $(\tilde{g}_1, \tilde{g}_2, \tilde{g}_2^{\gamma}, \cdots, \tilde{g}_2^{\gamma^m})$ and n tuples $(\tilde{A}_i, x_{i1}, x_{i2}, \cdots x_{im})$, $\forall i \in [1,n]$, where $x_{ik} \in \mathbb{Z}_p^*$ $\forall i \in [1,n], k \in [1,m]$, $\tilde{g}_2 \in \mathbb{G}_2$, $\tilde{g}_1 = \psi(\tilde{g}_2)$ and $\tilde{A}_i = \tilde{g}_1^{1/\prod_{k=1}^{m}(\gamma + x_{ik})}$, 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 $x_* \neq x_{ik}$ $\forall i \in [1,n], k \in [1,m]$ 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 $q = (n+1)m$.*

PROOF. Algorithm $\mathcal{B}$ is given a q-BSDH instance represented by $(g_1, w_0, w_1, \cdots, w_q)$, where $w_j = g_2^{\gamma^j}$, $\forall j \in [0,q]$. $\mathcal{B}$ sets $q = (n+1)m$. The objective of $\mathcal{B}$ is to produce a BSDH pair $(e(g_1,g_2)^{1/(\gamma+d)}, d)$ for some $d \in \mathbb{Z}_p^*$. For this, $\mathcal{B}$ creates the following framework to interact with $\mathcal{A}$.

Setup: $\mathcal{B}$ does the following.

1. Select nm values: $x_{ik} \xleftarrow{R} \mathbb{Z}_p^*$, $\forall i \in [1,n], k \in [1,m]$.
2. Define $\pi_i = \prod_{k=1}^{m}(\gamma + x_{ik})$, and $f(\gamma) = \prod_{i=1}^{n}\pi_i = \sum_{j=0}^{nm}\alpha_j\gamma^j$, where $\alpha_0, \alpha_1, \cdots, \alpha_{nm} \in \mathbb{Z}_p^*$ are the coefficients of the polynomial f with variable γ.
3. Compute $\tilde{g}_2 = g_2^{f(\gamma)} = \prod_{j=0}^{nm} w_j^{\alpha_j}$, and $\tilde{g}_1 = \psi(\tilde{g}_2)$.
4. Compute $\tilde{w}_k = \tilde{g}_2^{\gamma^k} = \prod_{j=0}^{nm} w_{j+k}^{\alpha_j}$, $\forall k \in [0,m]$.
5. Define $f_i(\gamma) = f(\gamma)/\pi_i = \prod_{j=1, j\neq i}^{n}\pi_j = \sum_{j=0}^{nm-m} a_j\gamma^j$, where $a_0, a_1, \cdots, a_{nm-m} \in \mathbb{Z}_p^*$ are the coefficients of the polynomial f_i.
6. Calculate $\tilde{D}_i = \tilde{g}_2^{1/\pi_i} = g_2^{f_i(\gamma)} = \prod_{j=0}^{nm-m} w_j^{a_j}$, and $\tilde{A}_i = \psi(\tilde{D}_i)$.
7. Send $(\tilde{A}_i, x_{i1}, x_{i2}, \cdots, x_{im})$, $\forall i \in [1,n]$, and $(\tilde{g}_1, \tilde{w}_0, \tilde{w}_1, \cdots, \tilde{w}_m)$ to $\mathcal{A}$.

Note that with this information, $\mathcal{A}$ or $\mathcal{B}$ can compute nm tuples $(\tilde{A}_i, \tilde{B}_i, \tilde{C}_{ik}, x_{ik})$ such that $e(\tilde{A}_i, \tilde{B}_i) = e(\tilde{g}_1, \tilde{g}_2)$ and $e(\tilde{g}_1, \tilde{B}_i) = e(\tilde{g}_1^{\gamma}\tilde{g}_1^{x_{ik}}, \tilde{C}_{ik})$ in the following manner.

1. Define $\pi_i = \prod_{k=1}^{m}(\gamma + x_{ik}) = \sum_{k=0}^{m} b_k\gamma^k$, where $b_0, b_1, \cdots b_m \in \mathbb{Z}_p^*$ are the coefficients of the polynomial defined by π_i.
2. Compute $\tilde{B}_i = \tilde{g}_2^{\pi_i} = \prod_{k=0}^{m}\tilde{w}_k^{b_k}$.
3. Define $f_{ik}(\gamma) = \pi_i/(\gamma + x_{ik}) = \prod_{j=1, j\neq k}^{m}(\gamma + x_{ij}) = \sum_{j=0}^{m-1} c_j\gamma^j$, where $c_0, c_1, \cdots c_{m-1} \in \mathbb{Z}_p^*$ are the coefficients of the polynomial f_{ik}.
4. Compute $\tilde{C}_{ik} = \tilde{g}_2^{f_{ik}(\gamma)} = \prod_{j=0}^{m-1}\tilde{w}_j^{c_j}$.

Also, $\mathcal{A}$ or $\mathcal{B}$ can compute nm BSDH pairs $(\tilde{E}_{ik}, x_{ik})$ in the following manner.

$$\tilde{E}_{ik} = e(\tilde{A}_i, \tilde{C}_{ik}) = e(\tilde{g}_1, \tilde{g}_2)^{1/(\gamma + x_{ik})}.$$

Output: $\mathcal{A}$ outputs a forged 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 $x_* \neq x_{ik}$, $\forall i \in [1,n], k \in [1,m]$, such that $e(\tilde{A}_*, \tilde{B}_*) = e(\tilde{g}_1, \tilde{g}_2)$ and $e(\tilde{g}_1, \tilde{B}_*) = e(\tilde{g}_1^{\gamma}\tilde{g}_1^{x_*}, \tilde{C}_*)$.

Having received the forged tuple from $\mathcal{A}$, $\mathcal{B}$ generates a new BSDH pair in the following manner.

1. Define
$$E' = e(A_*, C_*) = e(\tilde{g}_1, \tilde{g}_2)^{1/(\gamma + x_*)} = e(\tilde{g}_1, g_2)^{f(\gamma)/(\gamma + x_*)}.$$
2. Rewrite $f(\gamma)$ as $f(\gamma) = (\gamma + x_*)f_d(\gamma) + d_*$ for some polynomial $f_d(\gamma) = \sum_{j=0}^{nm-1} d_j\gamma^j$, and constant $d_* \in \mathbb{Z}_p^*$. This means that
$$E' = e(\tilde{g}_1, g_2)^{f_d(\gamma) + d_*/(\gamma + x_*)}.$$
3. Compute $g_2^{f_d(\gamma)} = \prod_{j=0}^{nm-1} w_j^{d_j}$, and
$$\tilde{E} = \left(E'/e(\tilde{g}_1, g_2^{f_d(\gamma)})\right)^{1/d_*} = e(\tilde{g}_1, g_2)^{1/(\gamma + x_*)}$$
$$= e(g_1, g_2)^{f(\gamma)/(\gamma + x_*)} = e(g_1, g_2)^{f_d(\gamma) + d_*/(\gamma + x_*)}.$$
4. Calculate
$$E_* = \left(\tilde{E}/e(g_1, g_2^{f_d(\gamma)})\right)^{1/d_*} = e(g_1, g_2)^{1/(\gamma + x_*)}.$$

Hence, $\mathcal{B}$ returns the tuple (E_*, x_*) as the solution to the submitted instance of the BSDH problem. □

THEOREM 2. *In the random oracle model, suppose an algorithm $\mathcal{A}$ breaks the traceability of GSPR with advantage ϵ, after q_H hash queries and q_S signature queries, then there exists an algorithm $\mathcal{B}$ that breaks the q-BSDH assumption with advantage $(\epsilon/n - 1/p)^2/16q_H$, where $q = (n+1)m$.*

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

Setup: $\mathcal{B}$ is given a bilinear group pair $(\mathbb{G}_1, \mathbb{G}_2)$ with respective generators g_1 and g_2. $\mathcal{B}$ is also given $(w_0, w_1, \cdots w_m)$, where $w_k = g_2^{\gamma^k}$, $\forall k \in [0, m]$. Further, $\mathcal{B}$ is given (A_i, y_i), $\forall i \in [1, n]$. For each i, either $s_i = 1$ indicating that a valid key pair (A_i, y_i) generated using equations (1) and (2) is known, or $s_i = 0$ indicating that A_i corresponding to y_i is not known. We run $\mathcal{A}$ giving it $gpk = (g_1, w_0, w_1, \cdots w_m)$ and y_i, $\forall i \in [1, n]$. Note that y_i can be used to generate the alias tokens using equation (1).

Queries: $\mathcal{A}$ can query $\mathcal{B}$ about the following.

- *Hash queries*: $\mathcal{A}$ queries the hash functions H_z and H_g, and B responds with random values with consistency.
- *Signing queries*: $\mathcal{A}$ requests a signature of member i on message M. If $s_i = 1$, $\mathcal{B}$ responds with the signature σ using Sign algorithm with the private key (A_i, y_i). If $s_i = 0$, $\mathcal{B}$ selects $x_{ik}, \alpha, \beta, \delta$ to compute T_1, T_2, T_3 and T_4 and the SPK V as in equation (5). If the hash function causes a collision, $\mathcal{B}$ declares failure and aborts; otherwise, $\mathcal{B}$ responds with $\sigma = (x_{ik}, T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_\delta)$. We assume that the signing queries related to a signer does not exceed m.
- *Corruption queries*: $\mathcal{A}$ requests the secret key of member i. If $s_i = 1$, $\mathcal{B}$ adds i to U, and responds with (A_i, y_i); otherwise, $\mathcal{B}$ declares failure and aborts. With (A_i, y_i), $\mathcal{A}$ can compute alias tokens x_{ik}, $\forall k \in [1, m]$ using equation (1), B_i using equation (3), and C_{ik}, $\forall k \in [1, m]$ using equation (4).

Output: Finally, if $\mathcal{A}$ is successful, it outputs a forged signature σ^* on a message M^* using tuple $(A_{i'}, B_{i'}, C_{i'k}, x_{i'k})$. If $\mathcal{B}$ fails to find the signer i' in U, it outputs σ^*; otherwise, $\mathcal{B}$ identifies some $i' = i$. If $s_{i'} = 0$, $\mathcal{B}$ outputs σ^*; otherwise, $\mathcal{B}$ declares failure and aborts.

With the above framework, there can be two types of forger algorithms [6]. Type I forger forges a signature of the member who is different from all $i \in [1, n]$. Type II forger forges a signature of the member $i \in [1, n]$ whose corruption is not requested. $\mathcal{B}$ treats these two types of forgers differently. Note that using the technique of Lemma 1, with a q-BSDH instance $(\hat{g}_1, \hat{g}_2, \hat{g}_2^{\gamma}, \cdots \hat{g}_2^{\gamma^q})$, $\mathcal{B}$ can obtain $(g_1, g_2, g_2^{\gamma^k}, \cdots, g_2^{\gamma^m})$, and $(q - m)$ BSDH pairs. Moreover, any BSDH pair besides these $(q - m)$ pairs can be transformed into a solution to the original q-BSDH instance which means that the q-BSDH assumption is broken.

Type I Forger: From an instance of $(n+1)m$-BSDH, $\mathcal{B}$ obtains $(g_1, g_2, g_2^{\gamma^k}, \cdots, g_2^{\gamma^m})$, and n tuples $(A_i, x_{i1}, x_{i2}, \cdots x_{im})$. From these n tuples, $\mathcal{B}$ obtains n valid key pairs (A_i, y_i) by setting $H_z(y_i, k) = x_{ik}$, $\forall i \in [1, n], k \in [1, m]$. $\mathcal{B}$ applies the above framework to $\mathcal{A}$. The framework succeeds whenever $\mathcal{A}$ succeeds. Hence, $\mathcal{B}$ obtains the Type I forgery with the probability ϵ.

Type II Forger: From an instance of nm-BSDH, $\mathcal{B}$ obtains $(g_1, g_2, g_2^{\gamma^k}, \cdots, g_2^{\gamma^m})$, and $n-1$ tuples $(A_i, x_{i1}, x_{i2}, \cdots x_{im})$. From these $n - 1$ tuples, $\mathcal{B}$ obtains $n - 1$ valid key pairs (A_i, y_i) by setting $H_z(y_i, k) = x_{ik}$ $\forall i \in [1, n-1], k \in [1, m]$. These $n - 1$ pairs (A_i, y_i) are distributed among n indices. $\mathcal{B}$ sets $s_{i'} = 0$ for the unfilled entry at random index i'. $\mathcal{B}$ selects $A_{i'} \xleftarrow{R} \mathbb{G}_1$, and $y_i \xleftarrow{R} \mathbb{Z}_p^*$. $\mathcal{B}$ applies the framework to $\mathcal{A}$. The framework succeeds only if $\mathcal{A}$ never requests the corruption of member i', but forges a signature that traces to $A_{i'}$. The value of i' is independent of the views of $\mathcal{A}$, and hence $\mathcal{B}$ obtains the Type II forgery with probability at least ϵ/n.

$\mathcal{B}$ obtains another BSDH pair beyond the given nm BSDH pairs using the framework with Type I or Type II forger in the following manner, contradicting the BSDH assumption. $\mathcal{B}$ rewinds the framework to obtain two forged signatures on the same message, where the commitments are the same, but the challenges and responses are different. The probability of success in achieving this is at least $(\epsilon' - 1/p)^2/16q_H$ by the forking lemma [6, 18], where ϵ' is the probability that the framework on each forger succeeds. $\mathcal{B}$ extracts (A_*, B_*, C_*, x_*) encoded in the forged signatures [6]. Further, $\mathcal{B}$ obtains a BSDH pair from (A_*, B_*, C_*, x_*) using the technique discussed in Lemma 1. The framework is successful only if the extracted BSDH pair is not among the BSDH pairs created by $\mathcal{B}$. Therefore, $\mathcal{B}$ obtains a new BSDH pair with the probability $(\epsilon' - 1/p)^2/16q_H$.

Hence, we have shown that $\mathcal{B}$ can solve the $(n+1)m$-BSDH instance with probability $(\epsilon - 1/p)^2/16q_H$ using Type I forger, and the nm-BSDH instance with probability $(\epsilon/n - 1/p)^2/16q_H$ using Type II forger. Therefore, the pessimistic Type II forger proves the theorem. This implies that traceability is satisfied in GSPR in the random oracle model under the BSDH assumption. □

5.4 Revocation Correctness

In the following discussion, we analyze the correctness of the results generated by the revocation check algorithm, RevCheck. The revocation correctness depends on the cross correlation property of the alias codes since the revocation code is generated by summing over multiple alias codes. Here, we discuss two categories of codes from the existing literature which can be potentially used as alias codes—orthogonal codes and non-return-to-zero (NRZ) based random codes. Through analytical results, we show that orthogonal codes and random codes are both inadequate for use in GSPR. Hence, we propose a new type of codes which we refer to as *piecewise-orthogonal codes* which can be used as alias codes. With the use of piecewise-orthogonal codes, GSPR's RevCheck algorithm does not determine the revocation status of a private key with certainty, but instead with a certain probability. If an alias token has been revoked and its corresponding alias code has been included in the revocation code, then RevCheck's result is guaranteed to be correct. However, there is a possibility of a false alarm. Using an iterative algorithm, this probability can be decreased iteratively, a la the well-known Miller-Rabin primality test algorithm [19]. The details of the revocation check procedure and iterative algorithm are given in Section 5.4.1.

For analyzing the revocation correctness, we define the two hypotheses—$H_0 : x_{ik}$ has been revoked, and $H_1 : x_{ik}$ has not been revoked. Here, the probability of false negative/dismissal, P_{fd}, can be defined as the probability of erroneously determining that a given alias token has not been revoked when it has been revoked by the group manager. In RevCheck, P_{fd} is equal to the probability of $z < \tau$ when H_0 is true. Also, probability of false positive/alarm, P_{fa}, can be defined as the probability of the verifier erroneously determining that a given alias token has been revoked when it has not been revoked by the group manager. In RevCheck, P_{fa} is equal to the probability of $z \geq \tau$ when H_1 is true. Further, we suppose that the number of revoked private keys is represented by n_r, and each alias token (or each element

Table 1: The alias and revocation codes used in the example.

s_1	$+1$	-1	-1	$+1$	$+1$	-1	-1	$+1$
s_2	$+1$	$+1$	-1	-1	$+1$	$+1$	-1	-1
s_3	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	-1
s_4	$+1$	-1	-1	$+1$	$+1$	-1	$+1$	-1
s_5	$+1$	-1	-1	$+1$	$+1$	$+1$	-1	-1
$RC = s_1 + s_2$	$+2$	0	-2	0	$+2$	0	-2	0

in $\mathbb{Z}_p^*$) is represented by $b_p = 160$ bits. Note that the number of revoked alias tokens (i.e., $m \cdot n_r$) is equal to the number of revoked alias codes, and the length of the revocation code is equal to the length of an alias code (i.e., l).

Orthogonal Codes: Orthogonal or Walsh codes consist of codes with zero cross-correlation [7]. When the two codes are the same, the value of the cross-correlation is 1; otherwise, it is 0. If these codes are used as alias codes, we can set the threshold $\tau = 1$, and the revocation check procedure with $P_{fd} = 0$ and $P_{fa} = 0$ can be achieved. This means that if we use orthogonal codes as alias codes, GSPR would be able to satisfy the revocation correctness property with certainty. However, there are only l unique orthogonal codes of length l samples. This means that if orthogonal codes are indexed using the alias token x_{ik} which is represented by $b_p = 160$ bits, then the length of each alias code has to be $l = 2^{160}$ samples long! Hence, it is prohibitively costly in terms of storage and processing overhead to use completely orthogonal codes as alias codes.

NRZ based Random Codes: Random codes can be generated by NRZ encoding of a random sequence of bits, which means bit 0 is mapped to sample -1, and bit 1 is mapped to sample $+1$. As a result, the number of unique random codes with length b_p is given by 2^{b_p}. If the random codes are utilized as alias codes, we can generate an alias code of length, $l = b_p = 160$ samples, by NRZ encoding of an alias token, x_{ik}. Although the use of random codes (as alias codes) allows us to use compact alias codes, they have a critical drawback; use of random codes results in $P_{fd} > 0$. As a result of the random nature of the codes, there are inevitable false dismissals, which means there is significant possibility that the verifier would not be able to detect a revoked private key. This is untenable in PPA as this could be utilized by adversaries to bypass the revocation check.

As discussed above, orthogonal codes as well as random codes have critical drawbacks that limit their utility as alias codes. Hence, we propose a new type of codes that we call *piecewise-orthogonal codes*. The use of piecewise-orthogonal codes enables us to create alias codes that are compact and have a very desirable property—viz., $P_{fd} = 0$ and $P_{fa} > 0$. In other words, when we use piecewise-orthogonal codes, the probability of false dismissals is guaranteed to be zero, although the probability of false alarms is non-zero. Note that ensuring $P_{fd} = 0$ is much more important than $P_{fa} = 0$ from a security point of view. The former implies that a revoked alias token can detected by RevCheck with 100% certainty. In the next subsection, we provide details on how piecewise-orthogonal codes are used in probabilistic revocation.

5.4.1 Revocation with Piecewise-Orthogonal Codes

In GSPR, we utilize piecewise-orthogonal codes as alias codes for achieving probabilistic revocation. The piecewise-orthogonal codes are generated by concatenating multiple segments where each segment is an orthogonal code. To generate a piecewise-orthogonal code as an alias code, an alias token is divided into multiple segments, and an orthogonal code is generated corresponding to each segment. These orthogonal codes corresponding to the segments of the alias token are concatenated to form the complete alias code. In this way, the alias codes are piecewise-orthogonal.

Specifically, a set of 2^{b_s} orthogonal codes, denoted by $\mathbb{C}_s$, is generated using the technique discussed in [7], where each orthogonal code is of length 2^{b_s}. Note that an orthogonal code in $\mathbb{C}_s$ can be retrieved using a b_s-bit index. Further, each alias token $x_{ik} \in \mathbb{Z}_p^*$ of b_p bits is divided into d segments each of length b_s bits, such that $d \cdot b_s \leq b_p < (d+1) \cdot b_s$. The segments of the alias token x_{ik} are represented by $x_{ik,j}$, $\forall j \in [1, d]$. Further, $\forall j \in [1, d]$, $x_{ik,j}$ is utilized to generate b_s-bit index so that an orthogonal code $s_{ik,j}$ is chosen from $\mathbb{C}_s$. Finally, all the d orthogonal codes, $s_{ik,j}$, $\forall j \in [1, d]$, are concatenated to generate the alias code s_{ik}. The length of the resulting revocation code is $l = d \cdot 2^{b_s}$. The group manager declares the two public parameters $\mathbb{C}_p$ and F_c, such that the set of all possible alias codes $\mathbb{C}_p = \mathbb{C}_s^d$, and the mapping function $F_c : \mathbb{Z}_p^* \rightarrow \mathbb{C}_p$ is defined as segment-wise indexing as discussed above.

When the revocation code is generated using the Revoke algorithm, each segment of the revocation code is generated by summation of the corresponding segments of the revoked alias codes. Hence, the generated revocation code also has d segments, represented by RC_j, $\forall j \in [1, d]$. Note that due to the property of orthogonal codes, the cross-correlation of a revocation code's segment and an orthogonal code results in one of the two values—(1) 0 if the revocation code was not generated by the orthogonal code, or (2) an integral multiple of 1 if the revocation code was generated by the orthogonal code. Hence, the threshold τ is set to 1.

Having received a signature with alias token x_{ik}, the verifier can run RevCheck for each of the d segments. However, to minimize the computational overhead, the verifier only runs RevCheck for a segments. This means that RevCheck can be re-organized as follows.

RevCheck(RC, σ)

1. Set $j = 1$.
2. Generate a b_s-bit index from $x_{ik,j}$, and select an orthogonal code $s_{ik,j}$ from $\mathbb{C}_s$.
3. Compute $z = \frac{1}{2^{b_s}} s_{ik,j}^T \cdot RC_j$. If $z \geq 1$, output invalid; otherwise, output `valid`, and exit.
4. Set $j = j + 1$. If $j \leq a$, go to Step 2; otherwise, exit.

5.4.2 Example

We illustrate the revocation check procedure in GSPR through an example. The alias codes and the revocation code used in the example are given in Table 1. We assume that there are five 4-bit alias tokens represented by $x_1 = \{1111\}$, $x_2 = \{1010\}$, $x_3 = \{0101\}$, $x_4 = \{1101\}$ and $x_5 = \{1110\}$. Also, we assume that $\mathbb{C}_s$ contains $2^2 = 4$ orthogonal codes. The group manager generates the alias codes s_1, s_2, s_3, s_4, and s_5—corresponding to x_1, x_2, x_3, x_4, and x_5, respectively—by concatenating two orthogonal codes of length 4 samples. Suppose that the group manager needs to revoke alias tokens x_1 and x_2. Hence, the group manager computes the sample-by-sample addition of the alias codes s_1 and s_2. The resulting vector is the revocation code, represented by RC. The group manager provides the verifier with RC. In this scenario, if the verifier receives a signature with the alias token x_1, he runs two iterations

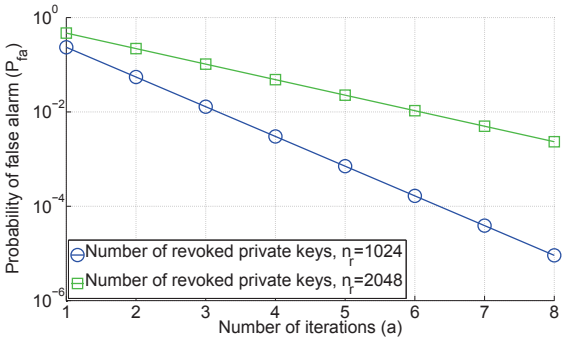

Figure 1: Probability of false alarm vs. number of iterations.

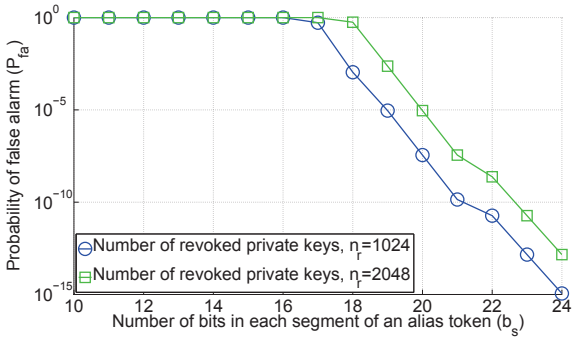

Figure 2: Probability of false alarm vs. number of bits in each segment of an alias token.

of RevCheck. In the first iteration, the verifier computes the cross correlation between the first segments, i.e., first 4 samples of s_1 and RC, represented by $s_{1,1}$ and RC_1, respectively. In the second iteration, the verifier computes the cross correlation between the second segments, i.e., second 4 samples of s_1 and RC represented by $s_{1,2}$ and RC_2, respectively. The cross correlation is computed by sample-by-sample multiplication of the alias code and the revocation code followed by the addition of all the products, and the resulting value is given by $\frac{1}{4}s_{1,1}^T \cdot RC_1 = 1$, and $\frac{1}{4}s_{1,2}^T \cdot RC_2 = 1$. Since the cross correlation of both the segments resulted in the value of 1, the verifier concludes that x_1 has been revoked. Using the same procedure, the verifier concludes that x_2 has also been revoked. On the other hand, if the verifier receives a signature with the alias token x_3, he will conclude that the alias token is valid because the cross correlation of $s_{3,1}$ with RC_1 is 0. Here, the RevCheck algorithm exits after the first iteration. Now, let us take a look at x_4. The cross correlation of $s_{4,1}$ and $s_{4,2}$ with RC_1 and RC_2 results in the values 1 and 0, respectively. Here, if the verifier makes a decision after only computing the correlation of the first segment, to decrease its computational overhead, he erroneously determines that x_4 has been revoked because this is an instance of a false alarm. However, after computing the correlation of the second segment, the verifier can conclude with absolute confidence that x_4 has not been revoked because $P_{fd} = 0$. Lastly, the cross correlation of $s_{5,1}$ and $s_{5,2}$ with RC_1 and RC_2 results in 1 and 1, respectively. Hence, if the verifier receives a signature with an alias token x_5, he erroneously concludes that x_5 has been revoked.

5.4.3 Discussions on the False Alarm Probability

With the proposed piecewise-orthogonal codes, the probability of false dismissal is zero, i.e., $P_{fd} = 0$. However, after checking a segments, the upper bound of the probability of false alarm (P_{fa}) can be computed to be

$$P_{fa} = \frac{(mn_r)^a \; 2^{b_p - ab_s} - mn_r}{2^{b_p} - mn_r} = \frac{n_t^a \; 2^{b_p} - mn_r}{2^{b_p} - mn_r} \approx n_t^a \quad (9)$$

where the ratio of the number of revoked alias tokens and the length of one segment of the revocation code is represented by $n_t = mn_r/2^{b_s}$. For $n_t < 1$, P_{fa} decreases by increasing a which is the maximum number of iterations or segments processed by the verifier before making a revocation status decision. Here, we assume that all revoked alias tokens have unique segments, and hence the above equation gives the upper bound of P_{fa}. Note that each alias token of b_p bits is unique; however each segment of an alias token, which is b_s bits long, is not necessarily unique.

If the verifier runs RevCheck for a iterations, then the length of the alias code that has to be processed is $l_r = a \cdot 2^{b_s}$, and $P_{fa} \approx n_t^{l_r n_t / mn_r}$. Note that the computational overhead for RevCheck is directly proportional to l_r. There is a tradeoff between P_{fa} and RevCheck's computational cost, and there are a number of different strategies for making an advantageous tradeoff. One possible strategy is to construct the revocation code in such a manner that minimizes P_{fa} for a given value of l_r and for a given number of revoked alias tokens (i.e., mn_r) by selecting an optimal value of b_s. Once the optimal value of b_s is computed, the corresponding n_t can be computed using the relation $n_t = mn_r/2^{b_s}$. This value can be readily derived as $n_t = \exp(-1) \approx 0.3679$. However, mn_r and 2^{b_s} are both integer values, and hence to minimize P_{fa}, the group manager needs to select b_s such that $\exp(-1)/2 \leq mn_r/2^{b_s} < 3\exp(-1)/2$.

As discussed above, the number of iterations (i.e., a) and the number of bits in each segment of an alias token (i.e., b_s) are adjustable parameters that directly impact P_{fa}. Figure 1 shows the impact of a on P_{fa} for a fixed value of $b_s = 19$. This figure suggests that the verifier can decrease P_{fa} at the cost of increasing the computational cost of performing RevCheck. Figure 2 illustrates the impact of b_s on P_{fa} when the verifier utilizes all of the d segments of the revocation code to check the revocation status of an alias token. In both figures, we fixed the values $m = 120$ and $b_p = 160$ bits.

5.4.4 Security Implications of the Alias Codes

There is a one-to-one mapping between an alias code and an alias token defined by F_c. Although the alias codes have a non-random structure, the alias tokens, which are embedded in the signature, are random numbers under the random oracle model. Hence, the use of alias codes should have no impact on the traceability property of GSPR, which is defined by Theorem 2.

6. PERFORMANCE EVALUATION

In this section, we evaluate the computational and communication overhead of GSPR, and compare GSPR's performance with two schemes in the prior art—the Boneh-Shacham (BS) scheme proposed in [6] and the Bichsel-Camenisch-Neven-Smart-Warinschi (BCNSW) scheme proposed in [3].

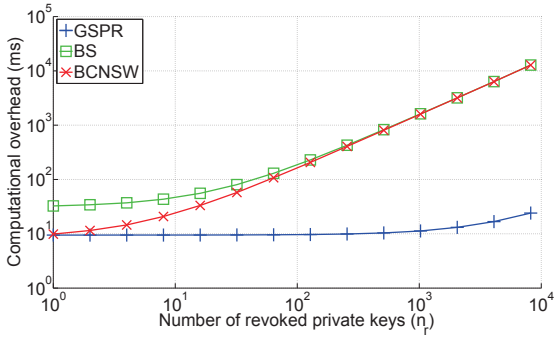

Figure 3: Computation overhead of verifying a signature vs. the number of revoked private keys.

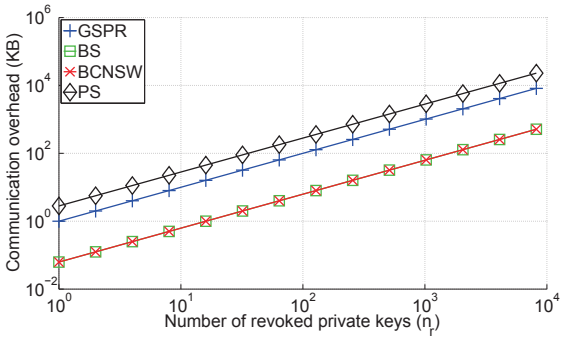

Figure 4: Communication overhead of transmitting the revocation list/code vs. the number of revoked private keys.

Table 2: Comparison of computationally expensive operations.

		Exp. in $\mathbb{G}_1/\mathbb{G}_2$	Exp. in $\mathbb{G}_T$	Bilinear map
GSPR	Sign	6	4	3
	SignCheck	2	5	4
	RevCheck	0	0	0
BS	Sign	5	3	3
	SignCheck	4	4	4
	RevCheck	0	0	$n_r + 1$
BCNSW	Sign	3	1	1
	SignCheck	0	2	5
	RevCheck	0	0	n_r

Table 3: Comparison of computational overhead (ms).

	Sign	SignCheck	RevCheck
GSPR	14.952	9.124	5.819
BS	15.417	15.378	1628.729
BCNSW	3.242	8.302	1592.019

Table 4: Comparison of number of elements communicated in the considered scenarios.

		Elem. in $\mathbb{Z}_p^*$	Elem. in $\mathbb{G}_1/\mathbb{G}_2$	Int.
GSPR	manager-signer	1	1	0
	signer-verifier	5	4	0
	manager-verifier	0	0	l
BS	manager-signer	1	1	0
	signer-verifier	5	2	0
	manager-verifier	0	n_r	0
BCNSW	manager-signer	1	3	0
	signer-verifier	2	3	0
	manager-verifier	0	n_r	0

Table 5: Comparison of communication overhead (bits).

	manager-signer	signer-verifier	manager-verifier
GSPR	672	2848	$5.03 \cdot 10^7$
BS	672	1824	$5.24 \cdot 10^5$
BCNSW	1696	1856	$5.24 \cdot 10^5$

In [13], Manulis et al. concluded that BS and BCNSW are two of the most practical group signature schemes in terms of being scalable to large networks. We assume that isomorphism is an identity map which means that $\mathbb{G}_1 = \mathbb{G}_2$. We assume symmetric 80-bit security level, which provides approximately the same level of security as an RSA signature with a modulus size of 1024 bits. In an elliptic curve cryptosystem, to achieve the same security strength, the length of an element in $\mathbb{Z}_p^*$, and $\mathbb{G}_1$ needs to be approximately equal to 160 bits [6]. Specifically, we utilize the "Type A" internal described in pairing-based cryptography (PBC) library available at [1]. The internal is constructed on a supersingular curve of the form $y^2 = x^3 + x$ over the field F_q for some prime $q = 3 \bmod 4$. In the internal, 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. For GSPR, we assume that the group manager distributes 120 alias tokens for each signer, and the verifier needs the probability of false alarm to be less than 0.01.

6.1 Computational Overhead

In this section, we compare the computational cost of GSPR with two benchmarks—viz., BS and BCNSW. We focus on three specific algorithms: Sign (signature generation algorithm), SignCheck (signature correctness checking algorithm), and RevCheck (revocation status checking algorithm). We focus on those algorithms because they need to be executed on-line in real time, and moreover they need to be performed by the signer and the verifier, who have limited computational capabilities compared to the group manager.

Firstly, we consider only the most computationally expensive operations—i.e., exponentiation (Exp.) in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$, and bilinear mapping. Here, since $\mathbb{G}_1 = \mathbb{G}_2$, the application of isomorphism is not considered. Table 2 provides the number of operations needed in each of the three algorithms for GSPR, BS and BCNSW. Note that in GSPR, the operations in Step 1 in the Sign algorithm are independent of the message to be signed or the random parameters, and hence, they can be pre-computed. Also, $\psi(w_1)$ and $e(g_1, g_2)$ can also be pre-computed. Further, in the RevCheck algorithm in GSPR, the computational cost of computing the cross-correlation between a revocation code and an alias code is l integer additions since the length of the revocation code is l with each element being an integer, and the alias code is a vector of $+1$s and -1s.

By using the PBC library, we implement the three algorithms for GSPR, BS and BCNSW, and measure their

running time on a PC platform with Intel(R) Core(TM)2 Duo CPU E8400 @ 3GHz. The measurements are obtained by averaging over 1000 runs of each algorithm. Table 3 provides their running times on the PC platform. Here, we assume that the number of revoked private keys is 1024, i.e, $n_r = 1024$. From Table 3, we can observe that there is no significant difference in the computation times of the three schemes when comparing their performance with respect to Sign and SignCheck. However, the difference between GSPR and the other two schemes in terms of the computational cost of RevCheck is significant. GSPR's RevCheck algorithm is more than two orders of magnitude more efficient than those of the other two schemes. Hence, when we consider the total signature verification time which includes the time needed to perform SignCheck as well as RevCheck, the running time in GSPR is significantly less than that in BS and BCNSW.

Figure 3 shows the computation time required to verify a signature versus the number of revoked private keys. We observe that with only a few thousand revoked private keys, the computation times for BS and BCNSW quickly grow to several seconds for verifying only one signature. In contrast, the growth rate of GSPR's computation time is much lower, which is primarily due to the computational efficiency advantage of GSPR's RevCheck.

6.2 Communication Overhead

We consider the three communication scenarios—between the group manager and the signer (manager-signer), between the signer and the verifier (signer-verifier), and between the group manager and the verifier (manager-verifier). In the first scenario, while joining the group, the group manager sends a secret key to the signer. In the second scenario, the signer sends a signature to the verifier. Lastly, in the third scenario, the group manager sends a revocation list/code to the verifier. Table 4 provides the number of elements (Elem.) of $\mathbb{Z}_p^*$, $\mathbb{G}_1$, $\mathbb{G}_2$ or integers (Int.) communicated in each of the three scenarios for GSPR, BS and BCNSW. Note that in GSPR, the alias tokens are generated by the signer using the secret key obtained from the group manager, and hence they do not need to be communicated.

Table 5 shows the required communication overhead of the three schemes for the three scenarios, assuming $n_r = 1024$. Results from the table indicate that GSPR's communication overhead is two orders of magnitude larger than those of the other two schemes when considering the manager-verifier scenario. Hence, we can conclude that GSPR makes an advantageous trade-off between computational overhead and communication overhead. This trade-off is advantageous because reducing the computational overhead is much more critical than reducing the communication overhead when considering scalability. Verifying a signature (which includes checking the revocation status of the private key) is an inherently on-line task which needs to be performed in real time, and it can be the primary performance bottleneck when the scheme is deployed in a large network. However, the greater communication overhead incurred by GSPR in the third (i.e., manager-verifier) scenario can be readily mitigated by pre-fetching the revocation code before the verifier needs to verify a given signature.

In Figure 4, we compare four schemes in terms of the communication overhead required to transmit the revocation list (for GSPR, it is the revocation code). The top-most curve

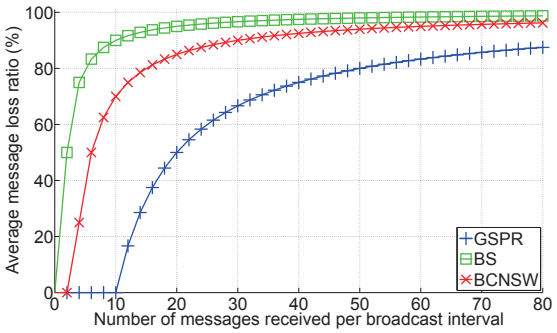

Figure 5: Average message loss ratio vs. number of messages received per broadcast interval.

is the curve for a pseudonym-based signature (PS) with Elliptic Curve Digital Signature Algorithm (ECDSA) with the public key size of 192 bits to achieve the 80-bit security level. For PS, we assume that the number of pseudonyms allotted to each signer is 120, and the group manager publishes public-key certificates of all the revoked pseudonyms in the revocation list. In Figure 4, we observe that although the communication overhead of GSPR is higher as compared to BS and BCNSW, it is still lower than PS.

7. USE OF GSPR IN DSRC APPLICATIONS

To illustrate the practical advantages of GSPR, in this section, we compare the signature verification performance of GSPR with two benchmarks (i.e., BS and BCNSW) for a specific type of applications, viz., vehicular network (VANET) safety applications. Since the allocation of the Dedicated Short-Range Communications (DSRC) spectrum in the 5.9 GHz band by the Federal Communications Commission (FCC), the automotive industry and the other stakeholders have been actively developing DSRC technologies, with a particular focus on vehicular safety applications.

In a typical safety application, each vehicle broadcasts beacon messages that contain information critical to safety, such as speed, direction of movement, acceleration, etc. The beacon messages need to be authenticated, but, at the same time, the privacy of the transmitting vehicle's driver must be protected [12]. Without such protection, adversaries can use the beacon messages to track the driver's movement or, worse yet, use them for more nefarious purposes. Hence, safety applications is one important application domain for privacy-preserving authentication techniques.

Typically, beacon messages are broadcast in intervals of 100 ms [12]. In high vehicular density scenarios, a given vehicle is expected to receive a large number of beacon messages within a broadcast interval, and each message needs to be authenticated before the arrival of the next message from the same transmitter. If the authentication of the current message cannot be finished before the arrival of the next message, then the current message must be discarded because it is considered to contain "stale" information. To measure the impact of the computational cost of signature verification on the performance of safety applications, we employ the average message loss ratio, which is defined as the ratio between the number of beacon messages discarded due to signature verification latency and the total number of

beacon messages received by a particular vehicle in a broadcast interval of 100 ms [12].

When simulating GSPR, we assume that each vehicle is on the road for 2 hours per day [20], and replaces its current alias token with a new one every minute, which equates to 120 alias tokens per day. The simulation results are shown in Figures 5 assuming $n_r = 64$. From this figure, we observe that GSPR's signature verification procedure is efficient enough to ensure acceptable performance for safety applications under reasonably-favorable conditions. In contrast, our results suggest that the computational burden of the verification procedures used by BS and BCNSW is too heavy for their use in vehicular safety applications.

8. RELATED WORK

The GS based schemes in the recent literature can be divided into two categories based on their revocation check procedures. In the first category of techniques, the revocation check procedure takes place at the signers [8, 11, 15]. The scheme proposed in [15] achieves constant signing and verification time at the cost of the public key of $O(\sqrt{n})$-size, where n is the total number of signers in the network. In [8], although the signing and verification have constant time along with constant-size group public key, the computational cost at the group manager grows with $O(n^2)$ which means that the group manager becomes the bottleneck. The scheme proposed in [11] achieves constant cost for signing and verification without significantly increasing the size of the public key. However, the length of each signature in [11] is significantly large, e.g., around 20 times that in [6]. While the schemes proposed in [8, 15] are secure in the random oracle model, the scheme proposed in [11] is constructed in the standard model.

In the second category of schemes, the revocation check procedure takes place at the verifier through verifier-local revocation (VLR) [3, 6, 16]. In these schemes, it is the responsibility of the verifier to check whether a signer has been revoked or not, by using the revocation list which contains the revocation tokens corresponding to the revoked private keys. However, the computational cost of revocation check procedure in these schemes increases linearly with the number of revoked private keys. These schemes are secure in the random oracle model.

9. CONCLUSION

In this paper, we proposed a novel privacy-preserving authentication scheme called Group Signatures with Probabilistic Revocation (GSPR). It is well known that revocation is the primary performance bottleneck of modern group signature schemes and that existing schemes do not scale well to large networks because of high computational cost of their revocation check procedures. By using the novel concept of probabilistic revocation, GSPR manages to significantly reduce the computational burden of the revocation check procedure at the cost of increased communication overhead. The negative impact of the increased communication overhead can be mitigated by pre-fetching the revocation code from the group manager before signature verification.

10. ACKNOWLEDGMENTS

This work was partially sponsored by NSF through grants 1228903, 1265886, 1314598, and 1431244; by NSFC through grant 61201245; and by the industry affiliates of the Broadband Wireless Access & Applications Center and the Wireless @ Virginia Tech group.

11. REFERENCES

[1] PBC: Pairing-based cryptography. `https://crypto.stanford.edu/pbc/`. Accessed: May 1, 2015.

[2] M. Bellare, D. Micciancio, et al. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology - EUROCRYPT*, volume 2656, pages 614–629. Springer Berlin Heidelberg, 2003.

[3] P. Bichsel, J. Camenisch, et al. Get shorty via group signatures without encryption. In *Security and Cryptography for Networks*, volume 6280, pages 381–398. Springer Berlin Heidelberg, 2010.

[4] D. Boneh, X. Boyen, et al. Short group signatures. In *Advances in Cryptology - CRYPTO*, volume 3152, pages 41–55. Springer Berlin Heidelberg, 2004.

[5] D. Boneh, B. Lynn, et al. Short signatures from the weil pairing. In *Advances in Cryptology - ASIACRYPT*, volume 2248, pages 514–532. Springer Berlin Heidelberg, 2001.

[6] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pages 168–177, 2004.

[7] E. H. Dinan and B. Jabbari. Spreading codes for direct sequence CDMA and wideband CDMA cellular networks. *Communications Magazine, IEEE*, 36(9):48–54, 1998.

[8] C.-I. Fan, R.-H. Hsu, et al. Group signature with constant revocation costs for signers and verifiers. In *Cryptology and Network Security*, volume 7092, pages 214–233. Springer Berlin Heidelberg, 2011.

[9] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in Cryptology - CRYPTO*, pages 186–194. Springer-Verlag, 1987.

[10] V. Goyal. Reducing trust in the PKG in identity based cryptosystems. In *Advances in Cryptology - CRYPTO*, volume 4622, pages 430–447. Springer Berlin Heidelberg, 2007.

[11] B. Libert, T. Peters, et al. Group signatures with almost-for-free revocation. In *Advances in Cryptology - CRYPTO*, volume 7417, pages 571–589. Springer Berlin Heidelberg, 2012.

[12] X. Lin, X. Sun, et al. GSIS: A secure and privacy-preserving protocol for vehicular communications. *Vehicular Technology, IEEE Transactions on*, 56(6):3442–3456, November 2007.

[13] M. Manulis, N. Fleischhacker, et al. Group signatures - authentication with privacy. Technical report, Group Signatures Study for BSI - German Federal Office for Information Security, 2012.

[14] A. Miyaji, M. Nakabayashi, et al. 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.

[15] T. Nakanishi, H. Fujii, et al. Revocable group signature schemes with constant costs for signing and verifying. In *Public Key Cryptography - PKC*, volume 5443, pages 463–480. Springer Berlin Heidelberg, 2009.

[16] T. Nakanishi and N. Funabiki. A short verifier-local revocation group signature scheme with backward unlinkability. In *Advances in Information and Computer Security*, volume 4266, pages 17–32. Springer Berlin Heidelberg, 2006.

[17] R. Pickholtz, D. Schilling, et al. Theory of spread-spectrum communications-A tutorial. *Communications, IEEE Transactions on*, 30(5):855–884, May 1982.

[18] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, pages 361–396, 2000.

[19] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.

[20] M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[21] P. P. Tsang, M. H. Au, et al. PEREA: Towards practical TTP-free revocation in anonymous authentication. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*, pages 333–344, 2008.