IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

IEEE RELIABILITY SOCIETY SECTION

Received July 25, 2020, accepted August 6, 2020, date of publication August 12, 2020, date of current version September 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3016041

A Privacy-Preserving Framework With Self-Governance and Permission Delegation in Online Social Networks

GUANGLAI GUO[1], YAN ZHU[1], (Member, IEEE), RUYUN YU[1], WILLIAM CHENG-CHUNG CHU[2], (Senior Member, IEEE), AND DI MA[3], (Senior Member, IEEE)

[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
[2]Department of Computer Science, Tunghai University, Taichung City 40704, Taiwan
[3]Department of Computer and Information Science, University of Michigan–Dearborn, Dearborn, MI 48128, USA

Corresponding authors: Yan Zhu (zhuyan@ustb.edu.cn) and William Cheng-Chung Chu (cchu@thu.edu.tw)

This work was supported in part by the National Key Technologies Research and Development Programs of China under Grant 2018YFB1402702, in part by the National Natural Science Foundation of China under Grant 61972032, and in part by the Beijing Municipal Bureau of Economy and Information Technology (Project: Development Research on Domestic and Foreign Big Data Standards).

ABSTRACT With the wide use of online social networks (OSNs), the problem of data privacy has attracted a lot of attention from not only the research community but also the general public. To meet the privacy needs of OSNs, we present a new framework for protecting information published through online social network websites through encryption by taking into account special features of OSNs. In this framework, autonomous private communities, called as *zones*, are set up by one or a set of mutually-trusted users collaboratively without any third party intervention. Sensitive information (i.e., posts, photos, etc.) within a zone can only be accessed by authorized members of the zone. A user joins a zone by obtaining a *permission* from an authorized zone member and uses it along with her private key to access contents inside the zone. One striking feature about our design of *permission* is that it is not secret information and thus can be left in the user's account in the OSN. Compared with prior work, this design of *public permission* greatly reduces user-side overhead on secret key management as a user only needs to maintain one secret key and use as many *public* permissions as she wants to access contents in different zones. Furthermore, our framework allows efficient access permission delegation and revocation. We develop a prototype to evaluate its computation performance in an acceptable level. Meanwhile, we prove that our construction is semantically secure against chosen plaintext attack, existential forgery attack and key forgery attack.

INDEX TERMS Security, social network, privacy protection, self-governance, key management.

Online social networks (OSNs) such as Facebook, LinkedIn, and Twitter are becoming increasingly popular and accessing these sites has become part of the daily routine of millions of users. These sites provide various tools and services that allow people to share content (such as contact information, personal tastes, photos, or viewpoints) and build communities that reflect their private and/or professional relationships in the real world [1], [2]. For example, people keep contact with friends and even make new friends through Facebook (http://www.facebook.com) or MySpace (http://www.myspace.com), build professional network and find job information from LinkedIn (http://www.linkedin.com), and so on.

As social network providers hold vast repositories of personal information, it raises great concern about potential user privacy violation. Although social network sites offer privacy controls that allow users to restrict whom their data can be viewed, they offer insufficient controls (both technically and legally) to restrict their own sharing of data with corporate affiliates or application developers [3]. User privacy still can be compromised in many possible ways such as users' poorly understanding of defaults or carelessness [4], accidental data release, intentional use of private data for marketing purposes [5], court order [6], and so on. Currently, OSN users have no control over their posted data. Also, it is unreasonable to

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun (Steven) Li.

expect OSN sites to provide any legal privacy protection for their users at time being [3].

The need for a mechanism to allow OSN users themselves to enforce access control of user-generated content has been identified by the research community [4], [7], [8]. Various techniques have been proposed to provide privacy protection against untrusted OSN providers. To avoid access by the OSN provider, Lockr [9] lets a user control its shared data by replacing the data with a link and storing the actual data at a third-party server. This approach shifts the requirement for trust on the OSN provider to the requirement of trust for the third-party sever. The work of [10] uses the concept of virtual private networks to build virtual OSNs which allow users to replace sensitive data with some pseudo information and store the real data on friends' machines. Since private data is stored in either a third-party server or friends' machines beyond the domain of the OSN, the OSN may not be able to access these data. However, in these schemes, users do not really have access control over their already posted data.

To protect user information and let users control the way how they want to share their data, one feasible solution would be for the user to encrypt the data she wants to share through the OSN platform, and give out the decryption keys only to authorized users. Unauthorized users, including the OSN providers, are not be able to decrypt since they do not have the decryption keys. This general idea actually have been widely adopted by many recent works [11]–[14]. However, existing solutions usually introduce a heavy overhead on OSN users for key distribution and management in order to meet some special needs required by the OSN setting, and thus do not scale well.

Some unique features of OSNs which require special handling in the design of an encryption-based access control scheme for the OSN setting include:

- *Dual User Roles*. An OSN user plays two roles as both a content *publisher* and *subscriber* in the OSN setting. An OSN user usually would like to share data with her friends - which is the primary reason why she joins the OSN. Reciprocally, her friends also would like to share their data with her. When a user publishes a content through the OSN platform for sharing, she acts as a *publisher*. When a user retrieves data shared by her friends in the OSN, she becomes a *subscriber* of her friends.
- *Unknown Recipients*. OSN users are not in charge of OSN memberships. A *publisher* may not know the set of recipients, or the *subscriber*s of her content, with whom her data will be shared beforehand. For example, Alice may post a message, in the encrypted form, on her friend Bob's wall. Alice's message could be accessed by Bob's friends if Bob allows his friends to see his wall. It is not possible or necessary for Alice to know the list of Bob's friends. Moreover, the number of subscribers a publisher may have might be a lot.
- *Efficient Subscriber-side Key Management*. A popular OSN user may have a lot of friends and thus needs to retrieve contents, as a *subscriber*, from many of her friends who publish and share data with her. That is, the number of publishers one may subscribe might be a lot. This requires efficient subscriber-side key management.
- *Multiple Private Groups*. The OSN platform usually provides a diversity of services including third-party applications such as Inbox, Wall, Blogs, Photos, the "like" page, etc., for users to share different contents. A user may want to share various contents with different sets of friends through different OSN services or applications.

To fit well into the OSN setting, an access control scheme should take into account the above special features of the OSN setting. Unfortunately, existing solutions often ignore one or more these special features. Moreover, they usually focus on designing access control schemes from the view point of the *publisher* and trying to minimize the overhead on the *publisher* side. They often ignore the fact that an OSN is also a *subscriber* who may subscribe to many *publisher*s. Thus, they inherently incur a heavy computation overhead on the *subscriber* side.

Schemes based on traditional cryptographic techniques such as symmetric or public encryption have limitations when dealing with multiple groups in OSNs since a *publisher* either needs to prepare multiple copies of encrypted data for each user in the group, or needs to know the identities of everyone to whom they give access [15]–[17]. To allow flexible access control of data sharing in multiple groups and minimize the computation overhead on the *publisher* side, many types of multi-party access control schemes [18]–[22] have been proposed in recent years. Pang and Zhang [21] define a new OSN model based on user-to-user relationship and public information, and introduce a variant of hybrid logic for formulating access control policies. Hu and Ahn [22] propose a multi-party authorization framework with authorized policy evaluation mechanism to enable collaborative management of shared data in OSNs. Schemes [23]–[28] based on attribute-based encryption (ABE) also have been posed in the OSN setting to provide the flexibility in controlling data sharing in multiple groups. Zhang proposes a privacy protection scheme [25] based on classified attribute encryption (PPSSN) to classify data owners and control access privilege with the mechanism of different close relationships accessing data with different degrees of accuracy. Luo [23] presents a hierarchical multi-authority and friend discovery scheme based on ciphertext-policy ABE (CP-ABE), which employs attribute subsets to achieve flexible fine-grained access control.

Although the use of ABE reduces the overhead of key management on the *publisher* side immediately when dealing with data sharing among multiple groups, it does no help on the *subscriber*-side key management. A *subscriber* essentially needs to maintain one (set of) decryption key for each *publisher* who shares data with her. A popular user may have hundreds or even thousands of OSN friends. Storing and managing hundreds or thousands of secret keys locally is by no means an easy job to end users. Especially, considering

that more and more users will access the OSN sites through their smartphones – given the increasing popularity of smartphones, storing and managing secret information on the smartphone platform not only poses more challenges but also may bring additional security risks due to factors such as: a smartphone may easily get lost or stolen; users tend to upgrade their smartphones to the latest model more frequently than to upgrade their desktop/laptop computers; the content of a smartphone is usually synchronized with that stored in the remote server provided by its service provider.

To achieve privacy preservation and secure communication in OSNs, some key management schemes have been proposed [29]–[34] in recent works. Muhammad and Mizanur [29] present a new and efficient centralized key management protocol to prevent Sybil attack and provide a secure communication service among users in OSNs. The core idea of this method is the existence of a 'roadblock' that any user intending to join a group must go through. Li and Wu [30] propose a secure chaotic maps-based group key agreement scheme to enhance the trustworthiness of the OSN systems. Jung and Nam [31] suggest an efficient key management scheme using Dynamic Identity-Based Broadcast Encryption (DIBBE), which realizes secure communication among users in decentralized social network. Yeh and Huang [32] introduce a security key agreement framework for simultaneous authenticating multiple users to improve the efficiency and security of peer-to-peer (P2P)-based OSNs.

We believe that a practical and effective access control and key management scheme in the OSN setting should provide the following properties:

Autonomy The security of a private OSN should be enforced by OSN users themselves without interventions from OSN providers or any other third parties.

Efficiency There should be minimum overhead for both *publisher*- and *subscriber*-side key management. The less secret information one has to store and maintain, the better.

Scalability There should be no restriction on the number of private groups that may form on the OSN, the number of users in a private group, and the number of groups one may subscribe to.

OUR CONTRIBUTIONS

To meet the privacy needs of OSN, we present *AutoZone*, a framework for protecting information published through OSN websites through encryption by taking into account the dual roles OSN users have and the special features of OSN communication. In *AutoZone*, autonomous private communities, called as *autozones* or simply *zones*, are set up by one or a set of mutually-trusted users collaboratively without any third party intervention. Sensitive information (i.e., posts, photos, etc.) within a zone can only be accessed by authorized members of the zone. A user joins a zone by obtaining a *permission* from an authorized zone member and uses the *permission* along with her secret key to decrypt content shared by other zone members. A permission is both user- and zone-dependent. Only the one who owns the permission can use it to decrypt contents in a specific zone. One user's permission is useless to the others. So a striking feature about our *permission* is that it is no longer secret information and thus can be stored remotely, for example, in the OSN, other than in the local storage on the user side. This novel design greatly reduces the overhead on secret key management on the subscriber side. In AutoZone, a user only needs to maintain one secret key. However, she can have as many *public* permissions as she can to access contents in different private zones. Moreover, AutoZone supports efficient permission delegation and provides a way to revoke the permissions of authorized members permanently or temporarily.

We summarize the key contributions of our work in this paper as follows.

- We present AutoZone, a system architecture for a private OSN. In this architecture, zone creators can collaborate to manage and maintain the privacy of contents inside the zone. There is no need for a centralized management server to help on key distribution and management and to monitor the behavior of all users.
- We propose an efficient access control scheme on top of the AutoZone architecture. We achieve minimum overhead on key management for subscribers through the design of *public permissions*. Our design also allows efficient access permission delegation and revocation.
- To prove the feasibility of our architecture, a prototype of AutoZone is implemented. Experimental results show that our construction can achieve the identified design goals for protecting privacy in OSN with acceptable performance. We also prove that our AutoZone scheme is semantically secure against chosen plaintext attack, existential forgery attack and key forgery attack.

ORGANIZATION

The rest of this paper is organized as follows. We introduce the basic concepts and the architecture of AutoZone in Section I. We articulate the key management scheme of AutoZone in Section II. We present the constructions of AutoZone functional modules in Section III. We analyze the security of AutoZone in Section IV. We report a prototype implementation of AutoZone and its performance evaluation in Section V. Finally, we concludes the paper in Section VI.

I. OVERVIEW OF AutoZone ARCHITECTURE

In this section, we introduce AutoZone, a private OSN architecture which allows secure sharing of data among community members via group key management and message encryption. We start with basic concepts in the AutoZone architecture.

A. AUTONOMOUS ZONES

In our private OSN setting, a *zone* refers to an exclusive community where contents (i.e., posts, photos, etc.) published inside the zone are restricted to be accessed only by

authorized zone members. By *autonomous*, we mean the security of the zone is governed exclusively by zone members without any intervention from the OSN provider or any other third party. By joining a zone, one gains the right to access contents posted by other members in this zone. The zone is an abstract concept which can support a variety of OSN applications/services:

A message inbox which allows an owner to send a message to a group of her friends simultaneously. The owner usually only needs to prepare one copy of the message and send the message once.

A blog which is a discussion or information site published on the World Wide Web consisting of discrete entries (''posts'') by the blog owner, usually a single individual. More recently ''multi-author blogs'' (MABs) have developed with posts written by large numbers of authors and professionally edited. Some sites, e.g., Twitter, allow bloggers to share thoughts and feelings instantaneously with friends and family.

A usenet network or newsgroup which many users can join. This is essentially equivalent to control of a blog for the administrators (called dealers).

A message whiteboard or bulletin board which allows the posting of messages. For example, the Facebook Wall is a peruser forum that features posts and comments from the user and her friends; the Facebook Photos application stores comments and tags for each picture and displays them to friends; and the Flickr photo management and sharing application allows each photograph has a page where members of the Flickr community can comment on photographs.

There is no restriction on the number of zones on can create, the number of applications/services a zone can support, and the number of zones one can join. A user (or a set of mutually trusted users) can set up different zones to meet different needs of data sharing. In the simplest case, a zone is set up to support data sharing of all content types among all friends. For example, in Alice's Facebook Account, Alice shares her posts, photos, videos, games, and so on with all her friends through various OSN services. A user can also set up a zone to specially support one single OSN service. For example, in Alice's Blog, Alice shares only her posts with her friends. Zones can also be set up for users to share different contents with different subsets of friends. For example, Alice can setup a zone to share photos taken in a group trip with a subset of her friends who were also in that trip.

B. AutoZone MEMBERSHIPS

For a specific zone, OSN users can be classified into four categories with different sets of privileges:

Kernel member (KM) A KM is an authorized zone member who is the owner or creator of the zone. A zone can be created by one or several mutually trusted OSN users collaboratively. So a zone can have one or multiple KMs. A KM has the rights to publish, delete, access or update contents released by other members of the zone.

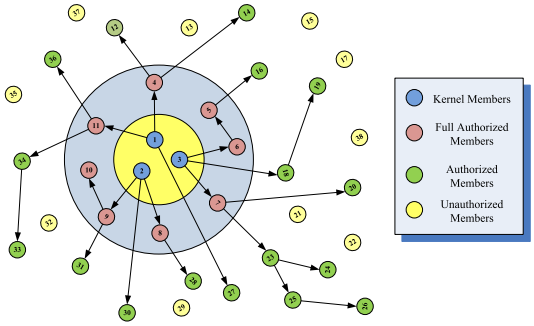


FIGURE 1. Privilege propagation in a zone where arrows are used to represent the direction of privilege delegation.

Full authorized member (FAM) A FAM is an authorized zone member who has full rights to publish and access resources in the zone, but do not have permissions to delete or update contents.

Authorized member (AM) An AM is an authorized zone member who can access resources in the zone with her access permission, but cannot publish any content.

Unauthorized user (UU) A UU is not an authorized zone member and does not have any permission to access resources published in the zone.

Apparently, among the three authorized zone memberships, KM is the most privileged while AM is the least privileged. AutoZone supports *privilege delegation* which allows a more privileged member to transfer her privilege to another user through a process called permission delegation. For security purpose, permission delegation should satisfy monotonicity. That is, access privileges are spread only from high-privileged members to low-privileged members. We provide an example in Figure 1 to describe the privilege propagation model used in AutoZone, as well as the relationships among four types of users.

A KM is able to assign her good friends as FAMs (e.g., Node 3 → Node 6 and 7), and assign her common friends as AMs (e.g., Node 3 → Node 18). Similarly, a FAM is also able to transfer her privilege (FAM) to her good friends (e.g., Node 6 → Node 5), and to assign her common friends as AMs (e.g., Node 7 → Node 20 and 23). Moreover, an AM is able to transfer her privilege (AM) to her (good/common) friends (e.g., Node 18 → Node 19).

The scale of privilege propagation should not be limited by the total number of users in the zone. Specifically, we have no restriction on the total number of FAMs and AMs in the same zone. However, the number of KMs in a zone usually should be limited due to the actual management complexity requirements. A user can have different memberships in different zones. Different memberships (KM, FAM, or AM) of a user in different zones are unrelated with each other.

We also note, it is technically possible using the ''delete'' and ''update'' operations to compromise the security of a

TABLE 1. Examples of key ownerships keys.

User ID	Private Key	Zone Key and Permission Key
0	sk_0	(gk_1)
1	sk_1	(gk_1); $(pm_{2(1)})$
2	sk_2	(gk_2)
3	sk_3	$(gk_1, pm_{1(3)})$; $(gk_2, pm_{2(3)})$
4	sk_4	$(gk_1, pm_{1(4)})$
5	sk_5	$(gk_1, pm_{1(5)})$
6	sk_6	$(gk_1, pm_{1(6)})$; $(pm_{2(6)})$
7	sk_7	$(pm_{1(7)})$; $(pm_{2(7)})$
8	sk_8	$(pm_{2(8)})$
9	sk_9	$(pm_{2(9)})$
10	sk_{10}	

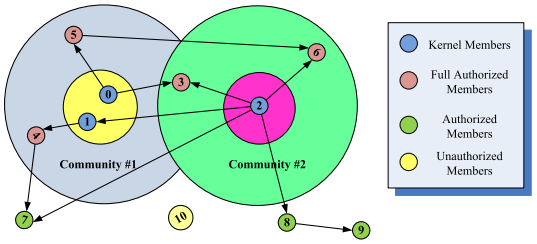


FIGURE 2. The key relation of our architecture.

zone. Hence, it is necessary to restrict maintenance operations of the zone only to KMs. So, it is critical to have an efficient authentication scheme to distinguish KMs from the others.

C. TYPES OF KEYS

In AutoZone, there are three types of keys used by authorized members of the zone.

- **User private key** sk. Every user in the OSN has a private key which is a user-specific secret. This key is generated and kept confidential by the user herself. The user registers the corresponding public key pk to the OSN.
- **Zone Key** gk. A zone key is a zone-wide secret. It is generated collaboratively by the zone KMs and built on the secrets of these KMs. The zone key is only given to KMs and FAMs.
- **Permission key** pm. Although we use the term "key", a permission key is not a secret. However, it is user- and zone-specific. It is generated from the zone key of a specific zone and the user's public key registered with the OSN. Only the owner of the permission can use it along with her private key to decrypt contents in that specific zone.

A permission key usually only allows one to decrypt content published by other zone members. To have the right to encrypt (or to publish), one needs the zone key. So for a specific zone, a FAM has both a zone key and a permission key while a normal AM only has a permission key. A KM does not need any permission to access contents in the zone. So a KM only has a zone key besides her own private key. We illustrate the ownerships of different keys among members in Figure 2 and Table 1.

Figure 2 shows two zones: Zone ♯1 and Zone ♯2, as well as 11 users (indexed from 0 to 10, i.e., u_i). Each user u_i has a private key sk_i. In addition, each user u_i has a zone key, or a permission, or both a zone key and a permission for each zone she belongs to depending on the membership type she has with that zone. In the example shown in Figure 2, u_0 and u_2 are two KMs in Zone ♯ 1 and Zone ♯ 2, respectively. They hold the zone-key gk_1 and gk_2, respectively. As a KM, they do not need permission to access the zone, at the same time, their private keys, sk_0 and sk_2, can guarantee that they pass the verification process for KMs. As a KM in Zone ♯1 and an AM in Zone ♯2, u_1 has a zone-key gk_1 and a permission $pm_{2(1)}$, where the subscript 2 and (1) of $pm_{2(1)}$ denotes the zone ID and the user ID, respectively. gk_1 and $pm_{2(1)}$ are independent to each other. As FAMs in both communities, u_3 has two key-pairs $(gk_1, pm_{1(3)})$ for Zone ♯ 1 and $(gk_2, pm_{2(3)})$ for Zone ♯2. Similarly, u_4 and u_5 have a key-pair of zone-key and permission from Zone ♯1, respectively. Note their permissions, $pm_{1(4)}$ and $pm_{1(5)}$, are different, and these permissions become valid only if they work together with the corresponding secret keys sk_4 and sk_5, respectively. The assignments of keys for other users are given in Table 1. Since u_{10} does not belong to any zone, she only has her private key.

D. OUR MODEL AND ARCHITECTURE

Our private OSN model could be built on existing social network platforms, such as Facebook, Orkut, etc, which usually allow developers to create "applications" to extend the types of information that can be stored, manipulated, and shared using social network interfaces. Fig. 3 depicts an OSN architecture based on existing Facebook. In this model, messages posted by end users are stored in database.

Client-side encryption is used to prevent unauthorized access of contents in a zone. As a middleware model, OSN platform is responsible for the interaction between end users and application providers. End users, consisting of KM, AM, FAM and UU, can easily establish contact with application providers by means of a URL on OSN platform. The platform firstly interprets the input data and related requests (HTTP Query) from the end users, then the interpreted input data and requests are transmitted to the application server through the network, the address of which is registered by the application developer on the platform in advance. Next, the application server performs the user input requests interpreted by the platform, perhaps containing operations on the database. The application server then provides the OSN platform with an output page containing of HTML and platform-specific markup (FBML), including scripts. After that, the OSN platform firstly interprets the output page, that is, converts platform-specific markup (FBML) into standard HTML and JavaScript. The interpreted user-recognizable output page (as HTML) is then handed over to the end users. In the above process, it is essential for client-side privacy to design a cryptographic module on ActiveX/JavaScript that is embedded into client's browser for decryption of output page.

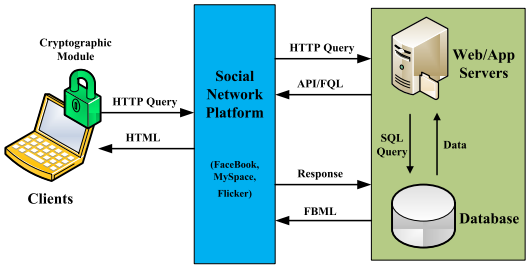


FIGURE 3. The private OSN architecture on Facebook.

In this architecture, the content publisher enforces access control through encryption and key management provided by AutoZone at client side. Based on the above-mentioned dataflow, AutoZone adopts five cryptographic function modules: *UserRegister*, *BuildZone*, *DelegatePermission*, *PublishContent*, and *AccessContent*, to control publishing and accessing contents in a zone as illustrated in Fig. 4.

- Each OSN user chooses a favorite label, generates her public/private key pair, and then use *UserRegister* module to register her label and public key on OSN platform.
- When a user or a set of mutually-trusted friends want to share data with others, an autonomous zone is setup through the *BuildZone* procedure. The creators (or KMs) get a zone key, which can be used to access, manage and maintain contents in this zone.
- When a user wishes to access a zone, her friends who has already been an authorized member of that zone, can delegate an access permission key (APK) to her by using the *DelegatePermission* module.
- When an authorized zone member wants to publish a content in the zone, she picks the zone key, invokes *PublishContent* algorithm to encrypt the content, and then transmit the encrypted data to the storage server; and
- Anytime a zone member can obtain the encrypted data from the sever and use the *AccessContent* module to retrieve the original data by using her private key and permission key.

II. AutoZone KEY MANAGEMENT

In this section, we articulate a concrete design for access control and key management based on the function modules in the AutoZone architecture. In our design, we intend to answer the following questions: How will a KM or several KMs define a zone and generate the zone key? How will authorized members delegate access privileges to others? How to encrypt a content for publishing and how to decrypt a content using permissions? And how does an untrusted third party (e.g. the OSN platform) can authenticate the KMs of the zone?

The *UserRegister* module is used for users to generate a private/public key pair without the intervention of the OSN. The security of AutoZone mainly relies on three modules, *BuildZone*, *DelegatePermission* and *Revocation*, to build a zone and grant/revoke access permissions without the help of OSN. The two modules, *PublishContent* and *AccessContent*, are used for publishing and accessing contents. Each module may employ one or more algorithms. In addition, a zone maintains a community member list (CML). Kernel members may update the CML to revoke certain members by the revocation algorithm. Notations used in the rest of paper are summarized in Table 2.

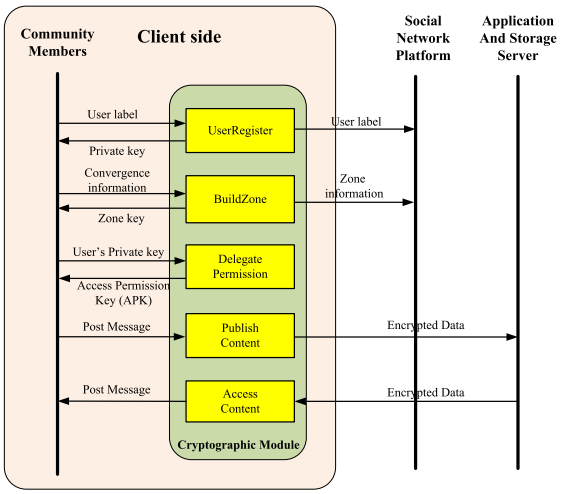


FIGURE 4. Cryptographic modules in AutoZone.

TABLE 2. Notations and symbols used in paper.

Term	Definition
id_i	user i's unique ID or label
sk_i	user i's private key
pk_i	user's i's public key
$pm_{j(i)}$	user i's permission to zone j
gk_j	zone key of zone j
$\mathcal{S}$	Set of KMs
$\mathcal{R}$	Set of revoked users
Σ	The zone convergence information
Setup(κ)	Initiate the global parameter list $param$ of the system by a security parameter κ
KeyGen($param$)	Generate the user private key sk_i based on the global parameter $param$
Register(id_i, sk_i)	Generate the public key pk_i from the private key sk_i
Converge($\mathcal{S}, \Psi$)	Generate convergence information Σ from the set of contributory shares Ψ from all KMs in $\mathcal{S}$, where Ψ is constructed on all sk_i in $\mathcal{S}$.
CKeyGen(Σ)	Build the zone key gk_j with a KM's private key sk_i and a convergence information Σ
CEncrypt $(sk_i, pm_{j(i)}, gk_j, M)$	Encrypt a content M by using a user's private key sk_i, a permission $pm_{j(i)}$, and a zone key gk_j
CDecrypt $(sk_i, pm_{j(i)}, C)$	Decrypt a ciphertext C by using a user's private key sk_i and a member's permission $pm_{j(i)}$
CVerify(M, C)	Verify the integrity of the resource M in the ciphertext C
Permission $(sk_i, pm_{j(i)}, gk_j, id_k)$	Generate the access permission $pm_{j(k)}$ of a zone by using a user's key sk_i, the permission $pm_{j(i)}$, a zone key gk_j and a target user's label id_k
Revocation $(sk_i, pm_{j(i)}, gk_j, \mathcal{R}, F)$	Revoke a set of users $\mathcal{R}$ by using any KM's private key sk_i, a permission $pm_{j(i)}$ and the zone key gk_j
KAuthenticate(A, B)	Authentication protocol between a KM A and a verifier B
FAuthenticate(A, B)	Authentication protocol between a FAM A and a verifier B

A. USER REGISTER

The system manager firstly generates the system parameter *param* by invoking *param* $\leftarrow$ *Setup*(κ), then makes it

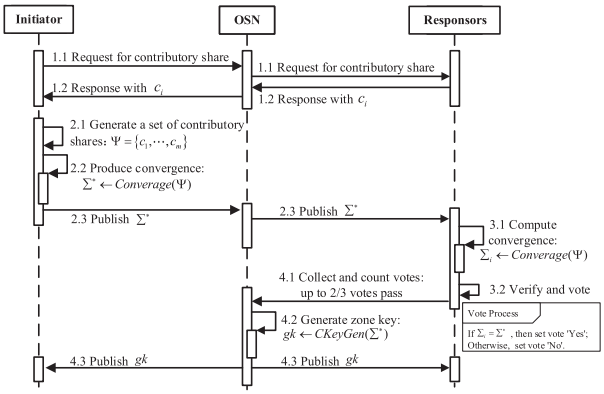


FIGURE 5. The workflow of zone key generation and distribution.

public. Based on *param*, each user u_i of the OSN chooses a unique ID id_i, generate its private sk_i by invoking $sk_i \leftarrow KeyGen(param)$. Then, the user registers herself with the OSN by sending $pk_i \leftarrow Register(id_i, sk_i)$ to the OSN. Here, the $Setup(\kappa)$ algorithm only needs to be invoked once and any knowledge of the user's private key can't be learnt by the system manager on registration.

B. BUILD ZONE

This module allows a set of mutually trusted users to collaboratively build a zone, and the OSN platform provides users with Web services of the zone (similar to the existing group service). Once the zone is built, these initial users are considered as kernel members (KMs) of the zone. Each KM will make a contributory share and generate a convergence information based on the contributory shares of all KMs in the zone. Furthermore, each of KMs should verify the correctness of the generated convergence information. Finally, the OSN platform builds a zone key from the confirmed convergence information.

Specifically, for a set of KMs $\mathcal{S} = \{u_1, \cdots, u_m\}$ in the zone, anyone in $\mathcal{S}$, can build a zone key with the help of the OSN platform (this KM is called the initiator, the others are called the responsor). The Fig. 5 shows the workflow of zone key generation and distribution, and the detailed process is described as follows:

Step 1. The initiator sends the request for the contributory shares to the OSN platform, then the OSN platform transmits the request to each KM u_i in $\mathcal{S}$. As a reply, each KM u_i responds to the initiator with her contributory share c_i via the OSN platform.

Step 2. The initiator uses Web script of the zone to generate a set of contributory shares $\Psi = \{c_1, \cdots, c_m\}$, and then produces a convergence information $\Sigma^* \leftarrow Converge(\mathcal{S}, \Psi)$ based on Ψ. Once this is done, the pair (Σ^*, Ψ) should be published to the Web page of the zone so that the responsors in the zone can obtain the result.

Step 3. After her contributory share c_i is confirmed in Ψ, each responsor u_i will generate her convergence information Σ_i in the same way as the initiator, and then verifies whether the value Σ_i is equal to the published Σ^*. If $\Sigma_i = \Sigma^*$, then the responsor votes 'Yes' to the zone page; Otherwise, it votes 'No'.

Step 4. The OSN platform collects and counts the votes, then checks whether the number of the approved responsors is up to 2/3 of the total votes. If it passes, the OSN platform generates the zone key gk by using the algorithm $CKeyGen(\Sigma^*)$ and publishes the zone key gk to the zone page; Otherwise, it reports the error to the page.

It is easy to see that the distribution process of zone key depends on the OSN platform as a result that the users do not need additional communication besides the OSN. So, malicious OSN platforms can interrupt the key distribution process, but the users need not to worry about their malicious behaviors since the mainstream OSNs are usually considered as honest-but-curious.

C. DELEGATE PERMISSION

Permission delegation refers to the concept that allows the permissions of a zone members to be transferred to her friends. By invoking $Permission(sk_i, pm_{j(i)}, gk_j, id_k)$, user u_i, be a KM or FAM, can delegate the "read" right of a zone to an external user u_k (who becomes an authorized member after getting the permission). In our scheme, only KMs and FAMs are required to have the right to perform permission delegation by using this algorithm for the sake of avoiding an unlimited delegation. If an external user wishes to get "write" permission of a zone, the user u_i in KM or FAM only needs to gives the zone key and the permission $pm_{j(i)}$ to her.

Permission generation can be implemented by the script of zone page in the client (its computational complexity is $(m^2+m-1)[E]$ for m KMs, which is introduced in Section V-B). The permission shall be stored at client-side in the form of key-value pair, the storage location of which might be cookie, registry, simple key-value database (such as MongoDB), JSON format files, etc. Considering the decryption process depends on both the permission and the user's private key, any malicious attacker can not decrypt the ciphertext if he does not have the user's private key but has the permission. Therefore, the permission need not to be stored in the encrypted form, but the security mechanism should be considered to protect the user's private key.

A key-value pair is usually denoted as *key* : *value*. The unique zone name can be generally taken as the *key*, and the *value* is used to store the permission (for our scheme in Section V-C, each permission is only one element under the group $\mathbb{G}$, and the size is approximately 3K bits). These key-value pairs will form a large array when a user joins too many zones. Fast retrieval technologies, such as Hash lookup table, can be applied to speed up the permission search.

D. PUBLISH CONTENT

The *PublishContent* is a process through which a KM or a FAM publishes a content to the zone. A member must hold a valid zone key gk_j to execute this process. Thus, AMs are not allowed to publish contents in the zone.

To prevent unauthorized publishing, an authentication protocol *FAthenticate*(A, B) is introduced to validate the identity of members. This protocol can avoid unauthorized users from uploading illegal ciphertexts to the zone. Once the authentication process is successful, the user is allowed to encrypt the content and then upload it to the content service provider (CSP) for sharing.

Algorithm 1 PublishContent(u_i, M)

1: $u \leftrightarrow OSN$: $b \leftarrow FAthenticate(u_i, OSN)$;
2: **if** b is true **then**
3: u_i: $C \leftarrow CEncrypt(sk_i, pm_{j(i)}, gk_j, M)$;
4: $u_i \rightarrow OSN$: C;
5: $OSN \rightarrow CSP$: upload(C);
6: **end if**

Using *PublishContent*(u_i, M), a user u_i in KM or FAM publishes a content M to the zone as follows: first of all, the member u_i performs *FAuthenticate* protocol to verify whether she is authorized through interactions with the OSN platform. If u_i is an authorized member, that is, she passes the authentication from the protocol *FAuthenticate*, she is allowed to encrypt the content M and submit the corresponding ciphertext C to the OSN platform. At last, the OSN platform transmits the ciphertext C to CSP.

E. ACCESS CONTENT

The AccessContent module allows a member to access contents in a private OSN. Given an encrypted content C derived from the OSN, any member u_i can use execute the algorithm *CDecrypt*$(sk_i, pm_{j(i)}, C)$ to decrypt it by her access permission $pm_{j(i)}$ and private key sk_i. This means anyone authorized in a private OSN can retrieve the encrypted content from the CSP.

Algorithm 2 AccessResources(u_i, C)

1: u_i: $M \leftarrow CDecrypt(sk_i, pm_{j(i)}, C)$
2: u_i: $b \leftarrow CVerify(M, C)$
3: **if** b is true **then**
4: u_i: Message is intact and output M
5: **end if**

In this module, we provide an efficient mechanism for the integrity check of message. Specifically, we make use of cryptographic Hash function to construct a verification algorithm *CVerify*(M, C) on the ciphertext C. After the decryption process of ciphertext C is done, the member can utilize the verification algorithm to verify whether the decrypted message M is intact. If the message M is tampered or corrupted, then the process abort; otherwise, the message M is indeed intact for the integrity check, and then the message can be displayed on the Web browser.

F. REVOCATION

The *Revocation* is a process through which a set $\mathcal{R}$ of members in a private OSN are excluded from accessing zone contents. The basic steps involved in the revocation is as follows: for a set $\mathcal{R}$ of revoked members (who can be determined by their public labels), by using the zone key along with her private key, the users in KM or FAM can invoke the algorithm *Revocation*$(sk_i, pm_{j(i)}, gk_j, \mathcal{R}, M)$ to encrypt the message M. This is only a temporary revocation in the sense, a user is only revoked regarding to this specific message. The user still can access other contents. In case of permanent revocation for an authorized member, the KM or FAM needs to add this authorized member into the revoked users list (CML) in the zone, and makes it public. While uploading a content into the zone, she simply sets the CML as the set $\mathcal{R}$ to encrypt the content. In the AutoZone scheme, we require that the number of revoked users must be strictly less than a threshold value (see Section III for more details). For improving revocation's capacity, we can easily increase the number of revoked users by increasing the threshold value during the zone key generation, at the sacrifice of increased performance overhead. But this kind of overhead could be compensated by the fast algorithm in Section V-A.

III. ALGORITHMS FOR AutoZone

In this section, we present the concrete constructions of algorithms used by function modules in the AutoZone architecture. Our constructions are built over groups with bilinear maps. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups of order q for a large prime q. A bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ between these two groups must satisfy the following properties: 1) Bilinear: $e(g^a, h^b) = e(g, h)^{ab}$ for all $g, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_q$. 2) Non-degenerate: $e(g, h) \neq 1$ unless g or h is the generator of $\mathbb{G}$. 3) Computable: there is an efficient algorithm to compute $e(g, h)$ for $g, h \in \mathbb{G}$.

We first present some intuition on the security of our constructions for the AutoZone cryptosystem. One challenge in building the AutoZone cryptosystem is how to allow a set of users to build a zone key collaboratively. We make use of *Lagrange interpolation polynomial* to solve this problem: Let $\mathbb{F}_q$ be a finite field. Given a set of m points $\{(x_0, y_0), (x_1, y_1), \cdots, (x_{m-1}, y_{m-1})\}$, where $x_i, y_i \in \mathbb{F}_q$ for $i \in [0, m-1]$ and all x_i are unique. Then there is a unique polynomial $f(x) \in \mathbb{F}_q[X]$ of degree at most $m-1$, that passes through all these m points, i.e. $f(x_i) = y_i$ for all $i \in [0, m-1]$. The polynomial $f(x)$ is defined as $f(x) = \sum_{i=0}^{m-1} y_i \cdot f_i(x)$, where $f_i(x) = \prod_{0 \leq j \leq m-1, j \neq i} \frac{x - x_j}{x_i - x_j}$, $i \in [0, m-1]$.

In AutoZone, a content is encrypted by a key derived from $f(0)$. As long as one know m points in the curve, one can recover the polynomial and thus the value of $f(0)$. In our design, the zone key provides $m-1$ points on the curve. To decrypt, one needs to find another point on the curve. In our design, the private key of a KM is a point on the curve so a KM does not need a permission to access the content. For a FAM or and an AM, the remaining point is provided by her private-key and permission to access the content. In fact, the user's private-key is a random point on any two-dimensional space. The permission is defined as the

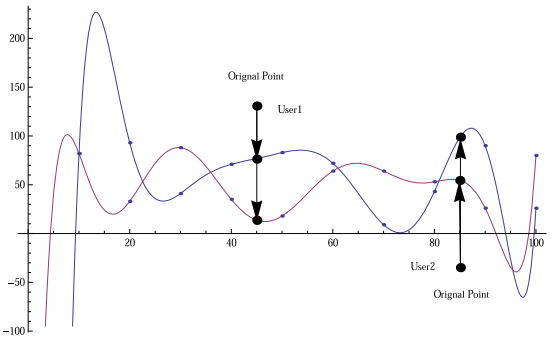


FIGURE 6. User private key and permission. Both curves shown in this figure can be used to build a zone key. A solid point outside the curve denotes a user's private key. The arrow from this outside point to a point on the curve denotes the user's permission.

y-axis distance from this random point to the curve (as shown in Figure 6). With both the private-key and permission, one can find the remaining point on the curve and thus recover $f(0)$. Of course, to ensure security, we will not let anyone to actually recover $f(0)$. However, such information is sufficient for an authorized user to perform decryption and access the content.

From the above description, it is easy to analyze the scalability which AutoZone can provide. For a large key space, e.g., suppose the key size is 320 bits, our scheme can support 2^{320} users and unlimited number of zones when using different generators in $\mathbb{G}$ to generate zone key. More importantly, the total number of users is unlimited in each zone and the total number of zones that one user can join is unlimited in terms of polynomial interpolation. The number of KMs is at most $m-1$ for a given polynomial with degree $m-1$. The degree of the polynomial can be adjusted according to actual application needs. For example, the degree should be small (from 10 to 50) for a small zone, e.g., a personal Blog; and it should be large (from 50 to 200) for a large zone. By using the efficient algorithm in V-A to calculate polynomial interpolation, we can ensure the performance of AutoZone when using a polynomial of high degree.

A collection of polynomial-time algorithms *(Setup, KeyGen, Converge, CKeyGen, CEncrypt, CDecrypt, CVerify, Permission, Revocation)* and two authentication protocols *(KAuthenticate, FAuthenticate)* are used to realize the function modules in the AutoZone architecture. We divide these algorithms into four categories, described in more detail in supplemental document.

IV. SECURITY ANALYSIS

In this section, we analyze the security of AutoZone against attacks, including chosen plaintext attacks, existential forgery attacks, and key-forgery attacks. Moreover, we provide a complete security analysis of the authentication protocols based on the interactive proof system (IPS). The proofs of theorems are provided in supplemental document.

A. SEMANTIC SECURITY FOR PRIVACY

We define the semantic security of our scheme using the following IND-CPA game. We say that an scheme ε is semantically secure (IND-CPA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger defined in the following IND-CPA game:

Setup The manager takes a security parameter κ to run the *Setup* algorithm, publish the system parameter *param*. The challenger and the adversary run *Register* algorithm respectively to get their private key sk_c, sk_a, then the challenger chooses a set of users, where the adversary is not included, runs the *CKeyGen* algorithm to generate the zone key gk and gives it to the adversary $\mathcal{A}$.

Phase 1 The adversary $\mathcal{A}$ issues encryption queries $X_1, \ldots, X_m$ to challenger. For each query X_i, the challenger responds by running algorithm $C_i = CEncrypt(X_i)$ to generate the ciphertexts and sends C_i to $\mathcal{A}$.

Challenge Once the adversary decides that Phase 1 is over, it outputs two equal length message M_0, M_1 on which it wished to be challenged. The only constraint is that M_0, M_1 don't appear in queries in Phase 1. The challenger picks a random bit $t \in \{0, 1\}$ and sets $C' = CEncrypt(M_t)$. It sends C' as the challenge to $\mathcal{A}$.

Phase 2 $\mathcal{A}$ issues more queries $X_{m+1}, \ldots, X_n$. The only constraint is that $X_i \neq M_0 \wedge X_i \neq M_1$. The challenger responds as in Phase 1.

Guess $\mathcal{A}$ outputs its guess $t' \in \{0, 1\}$ and wins if $t = t'$.

We refer to such an adversary $\mathcal{A}$ as an IND-CPA attacker. We define an adversary $\mathcal{A}$'s IND-CPA advantage in attacking our scheme ε as

$$Adv_{\varepsilon}^{cpa}(\mathcal{A}, \kappa) = \left| Pr[t = t'] - 1/2 \right|.$$

The presented scheme has the IND-CPA Security under the Decisional Bilinear Diffie-Hellman (BDH) assumption: Let $a, b, c, z \in \mathbb{Z}_n$ be chosen at random and g be a generator of $\mathbb{G}$, the decisional BDH Assumption is that no probabilistic polynomial-time algorithm $\mathcal{A}$ can distinguish the tuple $(g^a, g^b, g^c, e(g, g)^{abc})$ from the tuple $(g^a, g^b, g^c, e(g, g)^z)$ with more than a negligible advantage, that is,

Theorem 1: The proposed scheme is semantically secure against chosen plaintext attacks (IND-CPA) assuming the BDDH Assumption holds.

B. SECURITY AGAINST FORGERY ATTACKS

In the proposed scheme, an AM has only the read right and cannot publish a content into the zone, that is, she cannot forge a valid ciphertext to pass the member authentication process. We restrict that the forged ciphertext using the same random r as the one used in existing ciphertext. We prove the anti-forgery property by means of the following game Exp^{euf} between the challenger and adversary:

init The system manager takes a security parameter κ to run the *Setup* algorithm, publish the system parameter *pm*. The challenger and the adversary run *Register* algorithm respectively to get their private key sk_c, id_c; sk_a, id_a.

Setup The challenger generate a zone key gk, runs $Permission(gk, id_a, pm)$ to generate the access permission of $\mathcal{A}$ and give it to the adversary.

Learning At any time $\mathcal{A}$ may issue queries to challenger. $\mathcal{B}$ responds to each query X_i as follows:

1. if the query is a hash query for the random oracle H_i, compute the hash value and sends it to $\mathcal{A}$.
2. if the query is a encryption query for the challenger, $\mathcal{B}$ generates the ciphertext C_i.

Forge Once the adversary decides the above phase is over, it outputs a forged ciphertext C' on which it wished to be challenged. The only constraint is that C' don't appear in ciphertexts it received in above phase. If C' is a valid ciphertext, which means it the encryption of some file F' and can be correctly decrypted by the challenger, then the adversary win the game. The challenger outputs 1 if C' is a valid forgery ciphertxet, output 0 otherwise.

We define adversary $\mathcal{A}$'s advantage in forging ciphertext against scheme ε as $Adv_{\varepsilon}^{euf}(\mathcal{A}, \kappa) = Pr[Exp_{\mathcal{A}}^{euf}(\kappa) = 1]$, where the probability is taken over the random bits used by the challenger and the adversary.

The ciphertext of proposed scheme is secure under the hard assumption of computational Diffie-Hellman problem: Let $\mathbb{G}$ be a cyclic group of order p. Given $g, g^a, g^b \in \mathbb{G}$, output $g^{ab} \in \mathbb{G}$. We say that algorithm $\mathcal{A}$ has advantage ϵ in solving CDH in G if $Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}] \geq \varepsilon$ where the probability is over the random choice of generator $g \in \mathbb{G}$, the random choice of $a, b \in \mathbb{Z}_p^*$, and the random bits of $\mathcal{A}$.

Theorem 2: The proposed scheme is secure under Type-1 of forging ciphertext attacks for authorized members assuming the difficulty of Computational Diffie-Hellman (CDH) problem in $\mathbb{G}$. In the attack, the forged ciphertext using the same random r as the one in original ciphertext.

One of related problems to forgery ciphertext is the security of the gk zone key against collusion attack. In our scheme, the zone key is generated collaboratively by KMs, and keeps public to KMs and FAMs. So, any malicious attacker can obtain the zone key as long as he colludes with one or more traitors of KMs or FAMs. However, the AMs don't know the zone key, so she has no right to publish a content. Thus, we only analyze the case that some of AMs participate in collusion.

According to the *Permission* algorithm, the permission is defined as $pm' = g^{f(x')}/g^{y'}$, where $(x', g'^{y'})$ is the public key and $g'^{f(x')} = g'^{s} \cdot \prod_{i=1}^{m-1}(g'^{a_i})^{x'^i}$. For more than m traitors of AMs with their own private keys and permissions, they can employ the Lagrange interpolation method to obtain the exponential coefficients $\{g'^{s} = g'^{a_0}, g'^{a_1}, \cdots, g'^{a_{m-1}}\}$ of $g'^{f(x)}$ over $\{(x'_1, g'^{f(x'_1)}), \cdots, (x'_m, g'^{f(x'_m)})\}$, which builds an unique exponential polynomial $g'^{f(x)} = g'^{s} \cdot \prod_{i=1}^{m-1}(g'^{a_i})^{x^i}$. In the original scheme, these exponential coefficients (called the convergence information Σ) are derived from *Converage* algorithm taking the contributory shares $\Psi = \{(x, \Theta), (l_1, g'^{f(l_1)}), \cdots, (l_{m-1}, g'^{f(l_{m-1})})\}$ as input. However, the output of *Converage* algorithm depends on random choice of the integer $l_k (1 \leq k \leq m-1)$. This means that it is impossible for $g'^{f(x)}$ to build the zone key gk, if the random choice is unknown. Therefore, the zone key gk in our scheme is secure against collusion attack considering that more than m traitors of AMs can not figure out the zone key by sharing their private keys and permissions. This result is reached from the Theorem 2.

C. SECURITY AGAINST KEY ATTACKS

In our scheme the user's privacy key is the core secret which is used to perform all authorized behaviors. In order to ensure the security of communities, we require the unauthorized members can not forge the kernel member's privacy key in our scheme. To verify this requirement, we prove that an unauthorized user, whose have no right to access messages in a community, can't access resources by forging a kernel member's private key. The following game Exp^{fk} captures this property.

Setup The manager takes a security parameter κ to run the *Setup* algorithm, publish the system parameter pm. The challenger and the adversary run *Register* algorithm respectively to get their private key sk_c, sh_a. Then, the challenger chooses a random set of users, where the adversary is not included, runs the *CKeyGen* algorithm to generate the zone key gk, and runs *Permission* to generate the access permission. Finally, the challenger gives the permission μ to the adversary $\mathcal{A}$.

Learning During this phase the adversary $\mathcal{A}$ makes the queries described below to the challenger.

1. Encryption query: $\mathcal{A}$ issues encryption queries $X_1, \ldots, X_m$ to challenger. For each query X_i, the challenger responds by running algorithm *CEncrypt* to return the ciphertexts C_i to $\mathcal{A}$.
2. Decryption query: $\mathcal{A}$ issues decryption queries $C_1, \ldots, C_m$ to challenger. For each query C_i, the challenger runs algorithm *CDecrypt* and response with the resulting plaintext F_i.

Forge Once the adversary decides the above phase is over, it outputs a forged key sk' and sends it to the challenger. If the sk' satisfies the following two conditions, then it is called a valid forgery: (1) The challenger chooses a random ciphertext C encrypted by sk, it correctly decrypts using sk'; (2) The challenger chooses a random file F, encrypts it by sk', and could correctly decrypts with sk. The challenger outputs 1 if sk' is a valid forgery, output 0 otherwise.

We define the adversary $\mathcal{A}$'s advantage in this attack as $Adv_{\varepsilon}^{fk}(\mathcal{A}, \kappa) = Pr[Exp_{\mathcal{A}}^{fk}(\kappa) = 1]$. Key security of our scheme is guaranteed by the Strong Discrete Logarithm assumption: For every polynomial $Q(\cdot)$, every PPT $\mathcal{A}$, and all sufficiently large k, $Pr[\mathcal{A}(p, g, g^x) = x; 1 \leq x \leq p-1] < 1/Q(k)$, where the probability is taken over all primes p such that $|p| < k$, and coin tosses of $\mathcal{A}$.

Theorem 3: The above scheme is secure against forging key attack under the strong discrete logarithm assumption.

D. SECURITY OF AUTHENTICATE PROTOCOLS

Definition 1 (Secure Authenticate Protocol): Given the community, KAuthenticate(A, B) protocol is called secure authenticate protocol if A is a probabilistic algorithm, B is a determinant polynomial time algorithm and there exists polynomial $p_1(\cdot)$ and $p_2(\cdot)$ which satisfy the followings:

Completeness: For any kernel member's secret y, it holds $\Pr[\langle A(y), B\rangle(x) = 1] \geq 1 - 1/p_1(\cdot)$;

Soundness: For any non-kernel member's secret y^* and any non-kernel member A^*, it holds $\Pr[\langle A^*(y^*), B\rangle(x) = 1] \leq 1/p_2(\cdot)$;

Zero-knowledge: There exists a probabilistic polynomial-time algorithm S^* (called *simulator*) such that for every probabilistic polynomial-time algorithm D, for every polynomial $p(\cdot)$, and for all sufficiently large s, it holds

$$\left| \begin{array}{c} \Pr[D(x, S^*(x)) = 1] \\ -\Pr[D(x, \langle A(y), B^*\rangle(x)) = 1] \end{array} \right| \leq 1/p(s),$$

where, $S^*(x)$ denotes the output of simulator S on common input x and $\langle A(y), B^*\rangle(x)$ denotes the output of interactive protocol between B^* and $A(y)$ on common input x, i.e., for any input y, the ensembles $S^*(x)$ and $\langle A(y), B^*\rangle(x)$ are computationally indistinguishable.

Theorem 4: The protocol KAuthenticate(A, B) is a secure authenticate protocol in our scheme.

The security definition of FAuthenticate(A, B), which includes completeness, soundness and zero-knowledge properties, is similar to that of KAuthenticate(A, B). We have the following theorem.

Theorem 5: The protocol FAuthenticate(A, B) is a secure authenticate protocol in our scheme.

V. PERFORMANCE EVALUATION OF PROTOTYPE

A. EFFICIENT CALCULATION OF THE INVERSE OF VANDEMONDE MATRIX

In algorithm Converge, one needs to compute the inverse of Vandermonde matrix V, that is

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{pmatrix},$$

which is directly related to the efficiency of community-key generation and revocation mechanism. In our implementation, we use an efficient algorithm given by Mikkawy [35] to compute the inverse, with a time complexity of $O(n^3)$. This algorithm is able to improve the efficiency of Converge for a high-degree polynomial (with a larger threshold value).

Given Vandermonde matrix V, we define elementary symmetric function $\sigma_{i,j}^{(n)}$ in $x_1, x_2, \cdots, x_{j-1}, x_{j+1}, \cdots, x_n$. Let $\sigma_{1,j}^{(n)} = 1$ for $1 \leq j \leq n$ and

$$\sigma_{i,j}^{(n)} = \sum_{\substack{1 \leq r_1 < r_2 < \cdots < r_{i-1} < n \\ r_k \neq j, 1 \leq k \leq i-1}} x_{r_1} x_{r_2} \cdots x_{r_{i-1}}$$

Algorithm 3 Compute Elementary Symmetric Function

```
Set σ_{1,1}^{(1)} = 1
for i = 2 to n do
    σ_{i,1}^{(i)} = σ_{i-1,1}^{(i-1)} x_i
    for j = i − 1 to 2 do
        σ_{j,1}^{(i)} = σ_{j-1,1}^{(i-1)} x_i + σ_{j,1}^{(i-1)}
    end for
end for
```

TABLE 3. Time complexity of the algorithms in our scheme.

Algorithm	Time complexity
Converge	$m^2[E]$
CKeyGen	$(m^2 - 2m + 1)[E] + m[B]$
CEncrypt	$(m + 1)[E] + m[B]$
CDecrypt	$m[E] + 2[B]$
Permission	$(m^2 + m - 1)[E]$
Revocation	$(m^2 + m + 1[E] + 2(m - 1)t[E] + m[B])$

for $2 \leq i \leq n$ and $1 \leq j \leq n$. All of $\sigma_{i,j}^{(n)}$ construct an $n \times n$-dimension matrix $\sigma_{n\times n}$. The first column of $\sigma_{n\times n}$ can be calculated by calling the Algorithm 3.

The elements in the remaining $n - 1$ columns of $\sigma_{n\times n}$ can be obtained by using

$$\sigma_{i,k}^{(n)} = \sigma_{i,1}^{(n)}\Big|_{x_k \to x_1}, \quad 1 \leq i \leq n,\ 2 \leq k \leq n,$$

i.e., replacing x_1 by x_k in the above algorithm to obtain the kth column $\sigma_{i,k}^{(n)}$, $1 \leq k \leq n$. Let $V^{-1} = (v_{i,j})$, $1 \leq i, j \leq n$, we compute

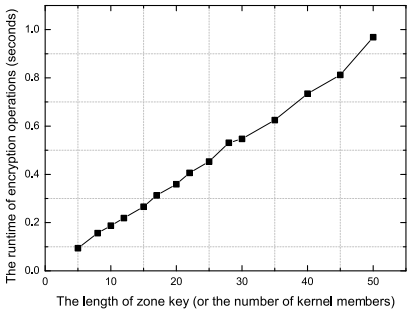
$$v_{i,j} = (-1)^{n+i} \frac{\sigma(n)_{n-i+1,j}}{\prod_{k=1,k\neq j}^{n}(x_j - x_k)}.$$

B. COMPUTATIONAL COMPLEXITY ANALYSIS

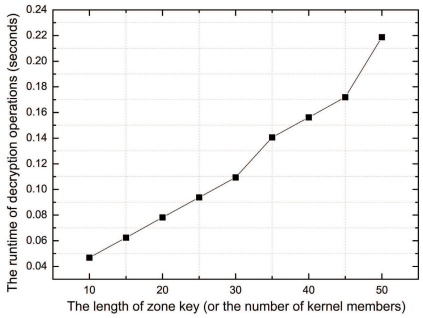
We analyze the computational complexity of major algorithms in terms of basic cryptographic operations. We denote the cost of one exponentiation operation in group $\mathbb{G}$ (or $\mathbb{G}_T$) as $[E]$, i.e., the time of computing g^x where $x \in \mathbb{Z}_q^*$, $g \in \mathbb{G}$ or $g \in \mathbb{G}_T$. We simply omit the algebraic calculation in $\mathbb{Z}_q$ and multiplication operations as they are very efficient. The cost of calculating bilinear map $e(\cdot, \cdot)$ is the most expensive one and we denote it as $[B]$. Let the number of kernel members is m and the number of revoked users in revocation is t. The cost of major algorithms in AutoZone is listed in Table 3.

C. SPACE COMPLEXITY ANALYSIS

We analyze the space complexity of major entities in our algorithms. Let $|\mathbb{G}|$ denote the size of elements in group $\mathbb{G}$. Similarly, the size of elements in group $\mathbb{G}_T$ and $\mathbb{Z}_q^+$ are denoted as $|\mathbb{G}_T|$ and $\left|\mathbb{Z}_q^+\right|$, respectively. The space complexity of five major entities, i.e., Σ, gk, hdr, sk and pm, is showed in Table 4. It is obvious that the storage overheads of three entities, the converenge information Σ, the zone key gk and the header of ciphertext hdr, are linear correlation



(a) The encryption operations for header generation.



(b) The decryption operations for header decryption.

FIGURE 7. The runtime of encryption and decryption under the different sizes of zone key.

TABLE 4. Space complexity of entities in our scheme.

Entity	Space Complexity	Overhead
Σ	$m\|\mathbb{G}\|$	60Kb (8KB)
gk	$m\|\mathbb{G}\| + (m-1)\|\mathbb{Z}_q^+\|$	64Kb (8KB)
hdr	$3\|\mathbb{G}\| + m\|\mathbb{G}_T\| + (m-1)\|\mathbb{Z}_q^+\|$	133Kb (17KB)
sk	$1\|\mathbb{Z}_q^+\|$	256b (32B)
pm	$1\|\mathbb{G}\|$	3Kb (384B)

with the number of zone's KM members m. For example, the entity $gk = \left\{g, (l_1, g^{f(l_1)}), \cdots, (l_{m-1}, g^{f(l_{m-1})})\right\}$ consists of m elements in $\mathbb{G}$ and $m-1$ elements in $\mathbb{Z}_q^+$, such that their corresponding storage overheads are $m\,|\mathbb{G}|$ and $(m-1)\left|\mathbb{Z}_q^+\right|$, respectively. So, the space complexity of gk is $m\,|\mathbb{G}| + (m-1)\left|\mathbb{Z}_q^+\right|$. In addition, the other entities, the private key sk and the permission pm, are of a fixed size shown in the Table 4.

In our scheme, we assume that the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is constructed on the curve $y^2 = x^3 + x \bmod p$ over the field $\mathbb{F}_p$ for some prime $p = 3 \bmod 4$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative groups of order q and the integer q is a prime factor of $p+1$. Under 128-bit security strength, it is required that the size of $\mathbb{F}_p$ is 1536 bits, and the size of (uncompressed) elements in $\mathbb{G}$ is 3072 bits, i.e., $|\mathbb{G}| = 3072$ bits. When the embedding degree k of the curve is 2, the size of (uncompressed) elements in $\mathbb{G}_T$ is required for 6144 bits, i.e., $|\mathbb{G}_T| = 6144$ bits. Correspondingly, the prime q need only to be 256-bit integer, i.e., $|\mathbb{Z}_q^+| = 256$ bits. According to the above settings, the storage overheads of five major entities are listed in Table 4 when m is 20. It is easy to see that the storage overheads of sk and pm are of a small and fixed size (32B and 384B, respectively), which are benefical to be stored at client-side. Meanwhile, the storage overheads of the other entities, Σ, gk and hdr, are all less than 17KB. Therefore, our scheme has low storage overheads in practical application.

D. EXPERIMENTAL RESULTS

By developing the experimental AutoZone prototype, we evaluate the performance of the AutoZone scheme from several aspects, including the zone key generation and encryption-decryption operation. We have implemented our basic scheme in C++. The machine we used runs Window Server 2003, has 512M RAM and 3.40 GH Intel Pentium D CPU. The hash function and symmetric cryptosystem we used are standard SHA and AES algorithms. We measure the performance of the AutoZone scheme based on this implementation. We set up our cryptosystem using bilinear pairings based on elliptic curve. In our scheme, the security parameter κ is 80-bits. we need the elliptic curve domain parameters over $\mathbb{F}_q$ with $|q| = 160$-bits.

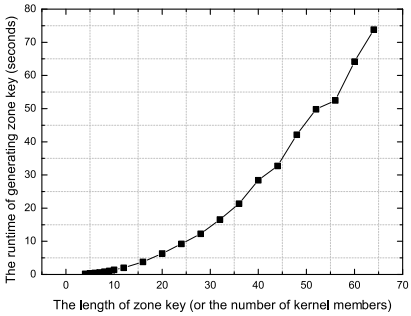


FIGURE 8. The runtime of generating zone key under the different lengths of zone key.

1) ZONE KEY GENERATION

Firstly, we focus on the computation cost for the *CKeyGen* algorithm. Fig. 8 shows the relationship between computation time and the length of zone key, where the the length of zone key is decided by the number of kernel members (and the number of revoked users). As we see, the computation time is roughly a quadratic function for the length of zone key. When the length of zone key is less than 10, the runtime of *CKeyGen* algorithm is less than 1.5 seconds. When the length comes up to 60, the runtime grows to 64 seconds. Since the number of kernel members is usually not large in a community, the scheme is feasible in practice.

2) ENCRYPTION AND DECRYPTION OPERATIONS

Secondly, we turn our attention to the performance of encryption algorithm. In order to improve the performance, our scheme uses a two-level encryption structure: the message is firstly encrypted by a session key via a symmetric encryption scheme, and then this session key is encrypted by our AutoZone scheme (called as the header of ciphertext). This structure solves the encryption problem of variable length message.

TABLE 5. Comparison between two existing OSN schemes and our scheme.

	flyByNight [36]	Persona [37]	Our scheme
Cryptosystem	ElGamal/Proxy encryption	PKC/ABE	Polynomial Interpolation
Autonomy	ElGamal managed by system manager; Proxy Encryption by application proxy	PKC managed by system manager; ABE managed by group creator	Full autonomy
Independence	Yes	Yes	Yes, a set of trusted users
Collaboration	No	No	Yes
Anonymous authentication	No	No	Yes
Revocation	No	No	Yes
Integrity checking	No	No	Yes
Relationship transitive	By group manager	From friend to friend	From friend to friend
Post message encryption	One-time by client-side; Each download by application proxy	One-time by client-side	One-time by client-side

In terms of this structure, the rumtime spent on encrypting a message can be divided into two parts: the runtime on generating the header and the runtime on encrypting the file by symmetric encryption. Here, we are only concerned about the former. In Fig. 7(a), we show the runtime of generating header under the different lengths of zone key, which is also denoted by the number of kernel members. It is easy to see that the computation time is approximately a linear function in the length of zone key. For a specific instance, when the file is 1KB, this figure shows that the runtime is less than one second, even when the length of zone key is grown to 50.

For the decryption operations, we are equally concerned about the decryption of the ciphertext header in the above-mentioned structure. In Fig. 7(b), we observe that the performance grow linearly with the length of zone key. As we can see, the time is 0.46 second for the length of zone key 10; and it is nearly 2.2 seconds for a longer zone key with 50 members. Thus, the latter is approximately 5 times as much as the former. But the runtime is considerably short under the different lengths of zone key.

3) PERFORMANCE COMPARISON WITH RELATED WORKS

Finally, Table 5 summarizes the comparison results between flyByNight [36], Persona [37], and our scheme. We can observe that our approach have following advantages: autonomy, collaboration, anonymous authentication, revocation, and integrity checking. These features could significantly mitigate privacy risks in using OSNs.

VI. CONCLUSION

In this paper, we presented AutoZone, a framework to secure social network data against untrusted OSN providers to meet the privacy needs of OSNs. AutoZone allows a set of mutually trusted uses to set up and manage a private zone where contents within the zone can only be accessed by authorized zone members. Its access permission delegation and revocation mechanisms provide flexibility in membership management. The concept of public permission greatly reduce the overhead for secret management for OSN users. Our proof-of-concept prototype clearly demonstrated practicality of AutoZone with manageable computation overheads.

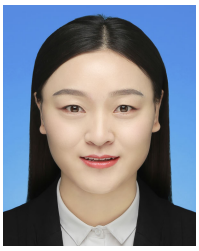
REFERENCES

[1] R. G. Pensa, G. Di Blasi, and L. Bioglio, "Network-aware privacy risk estimation in online social networks," *Social Netw. Anal. Mining*, vol. 9, no. 1, pp. 1–15, Dec. 2019.
[2] P. van Schaik, J. Jansen, J. Onibokun, J. Camp, and P. Kusev, "Security and privacy in online social networking: Risk perceptions and precautionary behaviour," *Comput. Hum. Behav.*, vol. 78, pp. 283–297, Jan. 2018.
[3] M. Lucas and N. Borisov, "FlyByNight: Mitigating the privacy risks of social networking," in *Proc. 7th ACM Workshop Privacy Electron. Soc.*, 2008, pp. 1–8.
[4] A. Acquisti and R. Gross, "Imagined communities: Awareness, information sharing, and privacy on the Facebook," in *Proc. Int. Workshop Privacy Enhancing Technol.*, 2006, pp. 36–58.
[5] B. Dybwad. (2010). *Facebook and Others Caught Sending User Data to Advertisers*. [Online]. Available: http://mashable.com/2010/05/20/facebook-caught-sending-user-data-to-advertisers/
[6] N. Andalibi and A. Forte, "Announcing pregnancy loss on Facebook: A decision-making framework for stigmatized disclosures on identified social network sites," in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, 2018, pp. 1–14.
[7] R. Gross and A. Acquisti, "Information revelation and privacy in online social networks (the Facebook case)," in *Proc. ACM Workshop Privacy Electron. Soc.*, 2005, pp. 71–80.
[8] B. Krishnamurthy and C. E. Wills, "Characterizing privacy in online social networks," in *Proc. 1st Workshop Online Social Netw. (WOSP)*, 2008, pp. 37–42.
[9] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman, "Lockr: Better privacy for social networks," in *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2009, pp. 169–180.
[10] M. Conti, A. Hasani, and B. Crispo, "Virtual private social networks," in *Proc. 1st ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, 2011, pp. 39–50.
[11] Z. Wang, Z. Ma, S. Luo, and H. Gao, "Enhanced instant message security and privacy protection scheme for mobile social network systems," *IEEE Access*, vol. 6, pp. 13706–13715, 2018.
[12] J. Su, Y. Cao, and Y. Chen, "Privacy preservation based on key attribute and structure generalization of social network for medical data publication," in *Proc. Int. Conf. Intell. Comput.* Nanchang, China: Springer, 2019, pp. 388–399.
[13] S. Jahid, P. Mittal, and N. Borisov, "EASiER: Encryption-based access control in social networks with efficient revocation," in *Proc. 6th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2011, pp. 411–415.
[14] A. De Salve, R. D. Pietro, P. Mori, and L. Ricci, "A logical key hierarchy based approach to preserve content privacy in decentralized online social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 1, pp. 2–21, Jan. 2020.
[15] Q. Wang, R. Feng, and Y. Zhu, "Verifiable random functions with Boolean function constraints," *Sci. China Inf. Sci.*, vol. 61, no. 3, pp. 039105:1–039105:3, Mar. 2018.
[16] M. Shehab, A. Squicciarini, G.-J. Ahn, and I. Kokkinou, "Access control for online social networks third party applications," *Comput. Secur.*, vol. 31, no. 8, pp. 897–911, Nov. 2012.
[17] A. K. Abdulla and S. Bakiras, "HITC: Data privacy in online social networks with fine-grained access control," in *Proc. 24th ACM Symp. Access Control Models Technol.*, May 2019, pp. 123–134.
[18] H. Hu, G.-J. Ahn, and J. Jorgensen, "Multiparty access control for online social networks: Model and mechanisms," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 7, pp. 1614–1627, Jul. 2013.
[19] B. Carminati and E. Ferrari, "Collaborative access control in on-line social networks," in *Proc. 7th Int. Conf. Collaborative Comput., Netw., Appl. Worksharing, CollaborateCom*, D. Georgakopoulos and J. B. D. Joshi, Eds. Orlando, FL, USA: IEEE, 2011, pp. 231–240.
[20] X. Zhang, Q. Zhou, C. Gu, and L. Han, "The location privacy preserving of social network based on RCCAM access control," *IETE Tech. Rev.*, vol. 35, no. 1, pp. 68–75, Dec. 2018.
[21] J. Pang and Y. Zhang, "A new access control scheme for Facebook-style social networks," *Comput. Secur.*, vol. 54, pp. 44–59, Oct. 2015.

[22] H. Hu and G.-J. Ahn, "Multiparty authorization framework for data sharing in online social networks," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*. Richmond, VA, USA: Springer, Jul. 2011, pp. 29–43.
[23] E. Luo, Q. Liu, and G. Wang, "Hierarchical multi-authority and attribute-based encryption friend discovery scheme in mobile social networks," *IEEE Commun. Lett.*, vol. 20, no. 9, pp. 1772–1775, Sep. 2016.
[24] Q. Huang, L. Wang, and Y. Yang, "DECENT: Secure and fine-grained data access control with policy updating for constrained IoT devices," *World Wide Web*, vol. 21, no. 1, pp. 151–167, Jan. 2018.
[25] L. Zhang, L. Li, E. Medwedeff, H. Huang, X. Fu, and R. Wang, "Privacy protection of social networks based on classified attribute encryption," *Secur. Commun. Netw.*, vol. 2019, pp. 1–14, Sep. 2019.
[26] W. Wei, S. Liu, W. Li, and D. Du, "Fractal intelligent privacy protection in online social network using attribute-based encryption schemes," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 3, pp. 736–747, Sep. 2018.
[27] Y. Yahiatene, D. E. Menacer, M. A. Riahla, A. Rachedi, and T. B. Tebibel, "Towards a distributed ABE based approach to protect privacy on online social networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–7.
[28] R. Xie, L. Yuan, G. Shi, Y. Wang, and C. Wang, "Attribute-based fine-grained extended access control mechanism for online social networks," in *Proc. Int. Conf. Comput. Intell. Syst. Netw. Remote Control (CISNRC)*, Shanghai, China, Dec. 2019, pp. 254–267.
[29] M. Al-Qurishi, S. M. M. Rahman, M. S. Hossain, A. Almogren, M. Alrubaian, A. Alamri, M. Al-Rakhami, and B. B. Gupta, "An efficient key agreement protocol for Sybil-precaution in online social networks," *Future Gener. Comput. Syst.*, vol. 84, pp. 139–148, Jul. 2018.
[30] C.-T. Li, T.-Y. Wu, and C.-M. Chen, "A provably secure group key agreement scheme with privacy preservation for online social networks using extended chaotic maps," *IEEE Access*, vol. 6, pp. 66742–66753, 2018.
[31] Y. Jung, Y. Nam, J. Kim, W. Jeon, H. Lee, and D. Won, "Key management scheme using dynamic identity-based broadcast encryption for social network services," in *Advances in Computer Science and Its Applications*. Danang, Vietnam: Springer, Dec. 2014, pp. 435–443.
[32] L.-Y. Yeh, Y.-L. Huang, A. D. Joseph, S. W. Shieh, and W.-J. Tsaur, "A batch-authenticated and key agreement framework for P2P-based online social networks," *IEEE Trans. Veh. Technol.*, vol. 61, no. 4, pp. 1907–1924, May 2012.
[33] S. Venkatesan, V. A. Oleshchuk, C. Chellappan, and S. Prakash, "Analysis of key management protocols for social networks," *Social Netw. Anal. Mining*, vol. 6, no. 1, p. 3, Dec. 2016.
[34] Y.-N. Liu, L. Harn, L. Mao, and Z. Xiong, "Full-healing group-key distribution in online social networks," *Int. J. Secur. Netw.*, vol. 11, nos. 1–2, pp. 12–24, 2016.
[35] M. E. A. El-Mikkawy, "Inversion of a generalized Vandermonde matrix," *Int. J. Comput. Math.*, vol. 80, no. 6, pp. 759–765, Jun. 2003.
[36] M. Lucas and N. Borisov, "FlyByNight: Mitigating the privacy risks of social networking," in *Proc. 5th Symp. Usable Privacy Secur. (SOUPS)*, 2009, pp. 1–8.
[37] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An online social network with user-defined privacy," in *Proc. SIGCOMM*, 2009, pp. 135–146.

GUANGLAI GUO received the M.S. degree in control theory from the University of Science and Technology (USTB), Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree with the School of Computer and Communication Engineering. His research interests include cryptography, access control, and network security.

YAN ZHU (Member, IEEE) was an Associate Professor with Peking University (PKU), China, from 2007 to 2012. He was a Visiting Associate Professor with Arizona State University (ASU) from 2008 to 2009, and a Visiting Research Investigator of the University of Michigan–Dearborn in 2012. He is currently a Professor with the School of Computer and Communication Engineering, University of Science and Technology Beijing (USTB), China. His research interests include cryptography, secure group computation, secure multi-party computing, and network security.



RUYUN YU received the M.S. and Ph.D. degrees from the University of Science and Technology Beijing (USTB), China, in 2015 and 2020, respectively. She was a Visiting Scholar as a Joint Ph.D. Student with the University of Michigan–Dearborn, from 2018 to 2019. Her research interests include group-oriented encryption and fine-grained access control.

WILLIAM CHENG-CHUNG CHU (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Northwestern University, Evanston, IL, USA, in 1987 and 1989, respectively. In 1992, he was a Visiting Scholar with Stanford University. From 1994 to 1998, he was the Dean of the Engineering College and an Associate Professor with the Department of Information Engineering and Computer Science, Feng Chia University. He was a Research Scientist with the Software Technology Center, Lockheed Missiles and Space Company, Inc. He is currently a Distinguished Professor with the Department of Computer Science, Tunghai University, Taichung City, Taiwan, where he had served as the Director of the Software Engineering and Technologies Center from 2004 to 2016 and as the Dean of Research and Development Office from 2004 to 2007. He has edited several books and authored or coauthored more than 100 refereed articles and book chapters, as well as participated in many international activities, including organizing international conferences, serving as the Steering Committee for the IEEE Computer Society Signature Conference on Computers, Software and Applications, the Asia-Pacific Software Engineering Conference, the IEEE International Conference on Software Quality, Reliability and Security, the International Symposium on System and Software Reliability, and the program committee of more than 70 international conferences. His current research interests include software engineering, artificial intelligence, and big data analytics. He was a recipient of special contribution awards in both 1992 and 1993 and a PIP Award in 1993 at Lockheed Missiles and Space Company, Inc. He is an Associate Editor of the IEEE TRANSACTIONS ON RELIABILITY, the *Journal of Software Maintenance and Evolution*, the *International Journal of Advancements in Computing Technology*, and the *Journal of Systems and Software*.

DI MA (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of California Irvine, Irvine, CA, USA, in 2009. She is currently an Associate Professor of computer science with the Department of Computer and Information Science, College of Engineering and Computer Science (CECS), University of Michigan–Dearborn, Dearborn, MI, USA. She is also the Interim Associate Dean for Graduate Education and Research and the Director of the Cybersecurity Center for Education, Research, and Outreach, CECS. Her research is supported by the National Science Foundation, the National Highway Traffic Safety Administration, the Air Force Office of Scientific Research, Intel, Ford, and Research in Motion. She was with the IBM Almaden Research Center in 2008 and the Institute for Infocomm Research, Singapore, from 2000 to 2005. She is broadly interested in the general area of security, privacy, and applied cryptography. Her research interests include wide range of topics, including connected and autonomous vehicle security, smartphone and mobile device security, radio frequency identification and sensor security, and data privacy. She was a recipient of the Tan Kah Kee Young Inventor Award in 2004, the Distinguished Research Award from the College of Engineering and Computer Science of the University of Michigan–Dearborn in 2017, and the 2018 Trevor O. Jones Outstanding Paper Award from SAE International.