Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Socialized policy administration

Zeqing Guo [a], *Weili Han* [a,*], *Liangxing Liu* [a], *Wenyuan Xu* [b], *Minyue Ni* [a], *Yunlei Zhao* [a], *Xiaoyang Sean Wang* [a]

[a] *Software School, and Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China*

[b] *Department of Electronic Engineering, Zhejiang University, Zhejiang Sheng, China*

ARTICLE INFO

Article history:
Received 4 August 2016
Received in revised form 26 January 2017
Accepted 6 March 2017
Available online 14 March 2017

Keywords:
Policy based management
Socialized policy administration
Mobile applications
Social computing
Android

ABSTRACT

With the rapid development of mobile applications and online social networks, users often encounter a frustrating challenge to set privacy and security policies (i.e., permission requests) of various applications correctly. For instance, in an Android system, it is hard for users, even programmers, to identify malicious permission requests (policies) when they install a third-party application. To simplify the task of policy management, in this paper, we propose a novel policy administration method where the policy settings from users' friends will be used as a key recommendation to guide policy administration, and the security of friends' privacy will be protected. We propose to let a user invite his or her friends to help with policy setting in applications, and we call such a method socialized policy administration (SPA for short). We designed two types of SPA: basic SPA and composite SPA. Both types of SPA are equipped with a privacy preserving mechanism that enables users' friends to help users without leaking the friends' preferences. In our prototype based on Telegram, i.e., one of the most popular instant messaging applications, we utilize partially homomorphic encryption cryptosystems to implement our framework. Based on the performance evaluation, SPA is able to configure almost all types of policies of current popular Android applications with a modest performance overhead.

1. Introduction

The rapid development of mobile applications and social network services raises the demand of user-friendly methods for policy administration, because these applications and services require their users to set various confusing yet obscured policies (i.e., approve or reject permission requests of applications). Unfortunately, users and even applications' developers (Fang et al., 2016) are usually unskilled at managing the applications (Enck et al., 2009; Barrera et al., 2010; Felt et al., 2011; Zhang et al., 2014; Fang et al., 2014). The common practice is that the people who are not good at security management might invite their friends and family members who are professional to help to set their applications. For example, one could invite his friends who major in Computer Science to install applications; an elder could ask his or her grandson to configure a smart phone. During these help session, it is common for the friends to show their configurations. Given that Social Network Services (SNSs for short) mimic relationships between

This paper is the extension version of "Z. Guo, W. Han, L. Liu, et al. SPA: Inviting Your Friends to Help Set Android Apps, In Proceedings of the 20th ACM Symposium on Access Control Models and Technologies (SACMAT 2015), June 1–3, 2015, Vienna, Austria: 221–231.

The paper is supported by NSFC (Grant No. 61572136, 61370080), the Shanghai Innovation Action Project (Grant No. 16DZ1100200), and the National Program on Key Basic Research (Grant No. 2015CB358800).

[*] *Corresponding author.*

E-mail address: wlhan@fudan.edu.cn (W. Han).

http://dx.doi.org/10.1016/j.cose.2017.03.005
0167-4048/© 2017 Elsevier Ltd. All rights reserved.

humans in reality, we extend this common practice to the cyberspace. In particular, a user may ask his or her friends to help with setting his applications via Social Network Services. This extension may appear to be simple and intuitive. However, such a method would not be acceptable if the privacy of a user's friends cannot be preserved, especially when a user consults them about sensitive settings, because if such a method cannot prevent personal information leakage, it can be used to gather friends' private information.

Existing methods of collaborative policy authoring (Wishart et al., 2010) or administration mechanisms (Han et al., 2014) do not protect friends' privacy. The prior work (Wishart et al., 2010) did aim to protect content privacy, e.g., how to protect the content of a user's posts while sharing among friends on an SNS, such as Facebook. However, prior work did not consider the contextual privacy introduced as friends interact with each other. In addition, CPA (Collaborative Policy Administration) (Han et al., 2014) utilizes settings of similar applications from friends to set a user's application without considering the privacy protection. To fill in the gap, we aim at designing a user-friendly policy administration method that does not leak user privacy.

This paper, therefore, presents *socialized policy administration*, whereby users ask their friends to help with privacy settings via SNSs without breaching friends' privacy. Our approach allows users with little knowledge of policy administration to set their privacy settings automatically. The main contributions of this paper are as follows:

- We design socialized policy administration (SPA for short) where a user can request his or her friends to help with setting up sensitive policies. We design two types of SPA: *Basic SPA* that treats each friend equally and *Composite SPA* that allows users to add weights to friends.
- We propose a privacy preserving method leveraging partially homomorphic encryption algorithms, which enable order comparison between two ciphertexts without decryption. In particular, comparison between ciphertexts supports *majority/minority* high level policies, and achieves a better performance of the merging algorithm for setting types (e.g., *Switch*, *Single Select*, *Multiple Select*) than the one using prior algorithms (Guo et al., 2015).
- We implement a prototype of *Composite SPA* on an Android client of Telegram, which is a popular instant messaging (IM) app with 100 millions monthly active users in February 2016 (TechCrunch, 2016). The composite SPA prototype allows users to request friends' settings and label a weight for each friend according to their professional knowledge about policy administration. Our evaluation of the prototype on Telegram illustrates its validity. The source code of prototype is uploaded on github[1].

Note that, although the policies of applications and services can be automatically set by SPA, users may view the settings as decision supports, and adjust the policies on their devices by themselves. As a result, professional users can also obtain useful references from their friends.

The rest of this paper is organized as follows: Section 2 introduces the background and describes the problem. Section 3 formally defines the SPA models. Section 4 describes the design and implementation of SPA. We then present our experimental process and evaluation results in Section 5. Next, we discuss the vulnerabilities of SPA and security of homomorphic encryption in Section 6. Section 7 introduces related work. Finally, Section 8 summarizes this paper and outlines our future work.

2. Background and motivation

2.1. *Homomorphic encryption*

Homomorphic encryption is a form of encryption that allows a set of computations to be carried out on ciphertext and generates an encrypted result which, when decrypted, matches the result of operations performed on the plaintext (Wikipedia, 2016). That is, A may encrypt a message m and send the ciphertext $\mathbb{E}(m)$ to B. B then take the ciphertext $\mathbb{E}(m)$ and evaluate a function $\mathbb{F}$ on $\mathbb{E}(m)$ to obtain the encrypted result $\mathbb{E}(\mathbb{F}(m))$. A decrypts the result, and obtain the expected functionality on m. Meanwhile B learns nothing about the data m.

Gentry showed the first fully homomorphic encryption scheme using lattice-based cryptography in 2009 (Gentry, 2009a; 2009b). Such a scheme allows one to compute arbitrary functions over encrypted data without the decryption key, i.e., given encryptions $\mathbb{E}(m_1), \dots, \mathbb{E}(m_t)$, one can compute a composite ciphertext that encrypts $\mathbb{F}(m_1, \dots, m_t)$ for any computable function $\mathbb{F}$. Although the fully homomorphic encryption (FHE) which supports an arbitrary function $\mathbb{F}$ on ciphertexts was proposed several years ago (Wang et al., 2015; Brakerski and Vaikuntanathan, 2011; Stehlé and Steinfeld, 2010; Van Dijk et al., 2010), its performance is hard to meet the requirements for a practical business service.

As a result, partially homomorphic cryptosystems are used in practice because they are faster yet provide partial homomorphic properties. These popular partially homomorphic cryptosystems include the following.

- **Paillier (Additive):** The Paillier cryptosystem, invented by Pascal Paillier in 1999, is a probabilistic asymmetric algorithm for public key cryptography (Paillier, 1999). The cryptographic algorithm generates a key pair, consisting of a public key and a private key. The public key is used to encrypt plaintext; whereas the private key is used to decrypt ciphertext.

The scheme is an additive homomorphic cryptosystem (Damgård and Jurik, 2001), which has the following property.

$$\mathbb{F}(\mathbb{E}(m_1), \mathbb{E}(m_2)) = \mathbb{E}(m_1 + m_2)$$

Here, $\mathbb{E}$ refers to encryption function, and $\mathbb{F}$ is a function defined by the partially homomorphic cryptosystem that cal-

[1] SPA prototype is accessible on https://github.com/SocializedPolicyAdministration.