PLOS ONE

RESEARCH ARTICLE

Obfuscating encrypted threshold signature algorithm and its applications in cloud computing

Yahong Li[1,2]*, Jianzhou Wei[3], Bin Wu[4], Chunli Wang[1], Caifen Wang[5], Yulei Zhang[4], Xiaodong Yang[4]

1 School of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou, Gansu, China, **2** School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China, **3** College of Science, Gansu Agricultural University, Lanzhou, Gansu, China, **4** College of Computer Science and Engineering, Northwest Normal University, Lanzhou, Gansu, China, **5** College of Big Data and Internet, Shenzhen Technology University, Shenzhen, China

* liyahong@mail.lzjtu.cn

Check for updates

OPEN ACCESS

Citation: Li Y, Wei J, Wu B, Wang C, Wang C, Zhang Y, et al. (2021) Obfuscating encrypted threshold signature algorithm and its applications in cloud computing. PLoS ONE 16(4): e0250259. https://doi.org/10.1371/journal.pone.0250259

Editor: Wenbo Shi, Northeastern University at Qinhuangdao, CHINA

Received: November 23, 2020

Accepted: April 3, 2021

Published: April 16, 2021

Copyright: © 2021 Li et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the manuscript and its Supporting information files.

Funding: This work was supported by the National Natural Science Foundation of China (61063041, 61901201, 61762058, 62002050, 61872060) and National Key R&D Program of China (2017YFB080200). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. The funders had no role in study design, data collection

Abstract

Current cloud computing causes serious restrictions to safeguarding users' data privacy. Since users' sensitive data is submitted in unencrypted forms to remote machines possessed and operated by untrusted service providers, users' sensitive data may be leaked by service providers. Program obfuscation shows the unique advantages that it can provide for cloud computing. In this paper, we construct an encrypted threshold signature functionality, which can outsource the threshold signing rights of users to cloud server securely by applying obfuscation, while revealing no more sensitive information. The obfuscator is proven to satisfy the average case virtual black box property and existentially unforgeable under the decisional linear (DLIN) assumption and computational Diffie-Hellman (CDH) assumption in the standard model. Moreover, we implement our scheme using the Java pairing-based cryptography library on a laptop.

Introduction

Cloud computing provides various data storage and services over a network [1]. Due to its many benefits, it collaborates with other promising technologies such as 5G networks [2, 3] and IoT [4, 5]. Meanwhile, more individual and corporate gradually outsource data storage or computation to the cloud for its cost saving and convenience. Despite various merits of cloud computing, however in practice, cloud servers are not entirely reliable [6–8]. Since if users directly delivery their data to cloud platforms, the important information in data will be leaked to cloud servers, which will lead to the exposure of users' privacy. Therefore, the concern is how to secure the data and rely on the services in cloud.

Obfuscation and cryptography are powerful tools that protect the data of users from a malicious/curious cloud server while preserving the services [9, 10]. When user chooses the cloud service to finish computation task without knowing the sensitive information of the task. In

and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

this case, the user data are obfuscated before they are forwarded to the cloud server. In a word, the cloud server can finish the computation tasks without sacrificing data privacy [11]. The related researches have addressed various security and privacy issues on data outsourcing. The works in [12–14] proposed cloud storage system in which data were obfuscated and encrypted. Sugumar et al. [15] proposed a confidentiality system named as SUG-DO (SUGUMARDigits Obfuscation) to enhance the security of data in cloud environment. Jin et al. [6] proposed an attribute-based data sharing scheme especially for resource-constrained mobile users in cloud computing. Recently, some new privacy preserving schemes [16–18]for other features have been reported to meet different auditing requirements in the literature.

Threshold cryptography is an essential distributed computational paradigm for enhancing the applicability and the security of public key schemes [19]. This approach was based on the work of Shamir [20], who proposed the definition of (t, n) threshold secret sharing scheme. Using Shamir's idea, (t, n)-threshold signature [21] separates private keys into n shares distributed to different users, t threshold or more share holders need cooperatively produce a signature. Most existing threshold algorithms rely on the trusted dealers (TDs) to distribute secrets, and more than that, it needs to be always trusted and safeguarded because it has the private key for all users, but they usually do not keep the confidentiality of the data against the cloud. To eliminate the third party, Pedersen [22] was the first to present threshold secret sharing scheme without any TDs. And then, based on the ideas similar to the protocol of Pedersen, Gennaro et al. [23] presented a secure distributed key generation for discrete log based cryptosystems (GJKR's DKG protocol) that enjoyed a full proof of security. Due to its distributed nature and the lack of a central authority, threshold cryptography becomes one of the most important tools in offering secure applications such as password protection [24] and cloud computing [25]. These studies make great contributions for protecting security of information systems and against various attacks. However, for nontrustable cloud, secret cryptographic keys are potentially vulnerable to attackers, the problem is related to ensuring proper protection of the outsourced computation task.

Program obfuscation is a very hot research topic in the field of practical application points of view, since program obfuscators perfectly conceal important information encoded into programs. A major breakthrough arrived with the work of Barak et al. [26] put forward the concept of program obfuscation into the area of cryptography, their work showed that the construction of generic obfuscation was impossible under the virtual black-box property. Many other impossibility results have been demonstrated in many situations [27, 28]. However, there are a few positive results for some functions in [29, 30]. Faced with the applications of cryptographic functionality, the first ever obfuscated re-encryption was mentioned in TCC'2007 by Hohenberger [31], a new security concept of average case virtual black-box property (ACVBP) was proposed. Succeeding their groundwork, Hada [32] proposed an obfuscator for encrypted signature scheme, and extended the definition of ACVBP, the algorithm was secure under the decisional linear assumption in the standard model. Consequently, several different functionalities and the corresponding obfuscators have been proposed. The research [33] showed a type of obfuscator for verifiably encrypted signatures. The obfuscation of encrypted group signature (EGS) was studied in [34], where the notion of ACVBP w.r.t $R(C)$ and $T(C)$ was defined. To provide secure authentication, Yang et al. [35], applied an obfuscator to anonymous authentication, the algorithm supported batch verification of authentication requests, realizing the improvement of efficiency. In order to make obfuscation application into cloud computing, Zhang et al. [36] proposed an obfuscator for all polynomial-size CNF circuits and used to cloud computing. Zhang et al. [37] proposed an obfuscator for encrypted verifiable encrypted signature, and modelled the application in electronic transactions. The obfuscation can achieve a series of applications, threshold signature is an attractive

service used in cloud computing, this paper focuses on achieving encrypted threshold signature, which designs an obfuscator to protect users' privacy. It should offer outsourcing computation without compromising data privacy.

However, the existing threshold cryptography mainly focuses on how to afford secure data for users, few works consider another requirement for the cloud application that needs to protect the sensitive data. In order to protect the privacy of the information sent from the user to the cloud, our work follows the idea of Hada's work and applies it to threshold signature setting. In this paper, we propose a secure obfuscation for encrypted threshold signature. The main contributions are as follows:

1. We propose an obfuscator that implements encrypted threshold signature (ETS) functionality, which can outsource the threshold signing rights of users to cloud server securely by obfuscation. Besides, this method can protect the sensitive leakage from the ETS program running on an untrusted sever.
2. We propose some security notions of ETS functionality and the corresponding obfuscator. Under the decisional linear assumption and computational Diffie–Hellman assumption, the proposed obfuscator satisfies the requirements of ACVBP and existentially unforgeability in the standard model.
3. We analyze the correctness of functionality preservation and polynomial slowdown. Meanwhile, the performance analysis of ETS functionality and the obfuscator are provided. Finally, we implement the proposed algorithms in a personal computer by using java pairing-based cryptography library.

The remainder of this paper is organized as follows. In section 2, we present some preliminaries including bilinear pairings, security problems and circuit obfuscators. In section 3, we present some build blocks will be used in our proposed schemes, then we propose an encrypted threshold signature scheme and the corresponding obfuscator based on linear encryption scheme and threshold signature. Section 4 analyzes the security and performance of our scheme from the perspectives of functionality preservation, ACVBP and existentially unforgeability. Section 5 presents our conclusion.

Preliminary

Bilinear pairings and security problems

In this section, we describe bilinear maps and hard problems [38]. Let consider two cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with the same prime order q, and let g is a generator of $\mathbb{G}$. A bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ need satisfy the following properties:

1. Bilinearity: For all $g, h \in \mathbb{G}$, and $a, b \in \mathbb{Z}_q$, $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.
2. Non-degeneracy: There exists $a, b \in \mathbb{Z}_q$, such that $\hat{e}(g^a, g^b) \neq 1$.
3. Computability: For all $g, h \in \mathbb{G}$, $\hat{e}(g, h)$ can be computed.

Definition 1. *The Decision Linear (DLLN) Problem is to decide whether $a + b = c$, given $u, v, h, u^a, v^b, h^c \in \mathbb{G}$ for unknown $a, b, c \in \mathbb{Z}_q$. The DLLN assumption states that, there is no PPT algorithm can solve the DLLN problem with non-negligible advantage.*

Definition 2. *The Computational Diffie-Hellman (CDH) Problem is that, given $g, g^x, g^y \in \mathbb{G}$ for unknown $x, y \in \mathbb{Z}_q$, it is hard to compute g^{xy}. The CDH assumption states that, there is no PPT algorithm can solve the CDH problem with non-negligible advantage.*

Circuit obfuscators

In this section, we briefly review some notations of circuit obfuscators used in this paper [32]. We use $\mathbb{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ to denote a class of probabilistic circles, here C_λ is the circuits in $\mathbb{C}$ of input length $l_{in}(\lambda)$. the notation $C \leftarrow C_\lambda$ denotes the generation procedure. PPT denotes probability polynomial time. Obf denotes an obfuscator. poly(λ) indicates the set of all polynomials of λ. We now provide definitions of statistical difference and preserving functionality.

Definition 3. [32] *The statistical difference between* $C_0(x)$ *and* $C_1(x)$ *is given by*:

$$\triangle(C_0(x), C_1(x)) = \frac{1}{2}\sum_{y\in\{0,1\}^{l_{out}(\lambda)}} |\Pr[o\longleftarrow C_0(x) : o = y] - \Pr[o\longleftarrow C_1(x) : o = y]|$$

Definition 4. *(Preserving Functionality)* [32] *A PPT machine Obf is a circuit obfuscator for a class of probabilistic circuits* $\mathbb{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$, *if for every probabilistic circuit* $C \in C_\lambda$, *the following holds*:

$$\Pr[C'\longleftarrow \mathrm{Obf}(C) : \forall x, \triangle(C(x), C'(x)) = 0] = 1.$$

Obfuscation of encrypted threshold signatures

Encrypted threshold signatures (ETS) functionality utilizes a threshold signature (TS) scheme, which was proposed in [21] and an asymmetric linear encryption scheme [39]. After that, we will give a detailed description of obfuscation.

TS signature

The TS signature scheme is a tuple of algorithms ∏ = (Setup, Share-Sign, Share-Verify, Combine, Verify) such that:

- **Setup**(*params*, λ, *k*, *n*): Takes as input a security parameter $\lambda \in \mathbb{N}$ and a pair of integers $(k, n) \in \mathrm{poly}(\lambda)$, such that $1 \le k \le n$, let $P = \{P_1, P_2, \ldots, P_n\}$ denote a set of *n* participants (users).
  1. Choose system parameter $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, q)$.
  2. $g_2, u', \mathbf{U} = (u_1, u_2, \cdots, u_n) \in \mathbb{G}^{n+2}$ are also generated by using GJKR's DKG algorithm [22], respectively.
  3. To generate public key, *n* users jointly generate user public key $g_1 = g^\alpha$ by using GJKR's DKG.
  4. Each user P_i broadcasts $g^{f(i)}$ for a random jointly generated degree $k-1$ polynomial $f \in \mathbb{Z}_q[X]$ such that $\alpha = f(0)$.
  5. User P_i gets the private key shares $\mathbf{SK} = (sk_1, sk_2, \cdots, sk_n)$ as $sk_i = g_2^{f(i)}$ for $i = 1$ to n. Verification keys $\mathbf{VK} = (vk_1, vk_2, \cdots, vk_n)$ as $vk_i = g^{f(i)}$ for $i = 1$ to n.
  6. Output the public key $p = (\mathbf{VK}, params, g_1, g_2, u', \mathbf{U})$, and each user is supplied with the private key share sk_i.
- **Share-Sign**(sk_i, *m*): To sign $m = m_1\, m_2 \cdots m_n \in \{0, 1\}^n$, using sk_i, choose $r_i \in \mathbb{Z}_q$, compute the signature share $\sigma_i = (\sigma_{i1}, \sigma_{i2}) = (sk_i(u'\prod_{i=1}^{n} u_i^{m_i})^{r_i}, g^{r_i})$.

- **Share-Verify**(p, m, i, σ_i): Given a signature share σ_i, and the verification key vk_i, the partial verification algorithm return 1 if $\hat{e}(\sigma_{i1}, g) = e(g_2, vk_i)\hat{e}(u' \prod_{i=1}^{n} u_i^{m_i}, \sigma_{i2})$, else return 0.
- **Combine**$(p, m, i, \sigma_i)_{i\in\Phi}$: For each $i \in \Phi$, where a subset $\Phi \subset \{1, 2, \cdots, n\}$ and $|\Phi| = k$. Let λ_i be the Lagrange coefficients so that $\alpha = f(0) = \sum_{i\in\Phi} \lambda_i f(i)$, compute the combined signature $(\overline{\sigma_1}, \overline{\sigma_2}) = (\prod_{i\in\Phi} \sigma_{i1}^{\lambda_i}, \prod_{i\in\Phi} \sigma_{i2}^{\lambda_i})$.
- **Verify** $(p, m, \overline{\sigma_1}, \overline{\sigma_2})$: Given signature $(\overline{\sigma_1}, \overline{\sigma_2})$, the receiver checks the equation $\hat{e}(\overline{\sigma_1}, g) = \hat{e}(g_2, g_1)\hat{e}(u' \prod_{i=1}^{n} u_i^{m_i}, \overline{\sigma_2})$. If the equation holds, outputs 1, otherwise outputs 0.

Linear encryption scheme

The linear encryption scheme consists of three algorithms Σ = (Key generation algorithm (KG), Encrypt algorithm(Enc), Decrypt algorithm(Dec)), the algorithms are described as follows:

- **KG**$(params)$: Parse system parameter $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, q)$, choose $a, b \in \mathbb{Z}_q$ as the private key sk_e, compute the encryption public key $pk_e = (pk_{e1}, pk_{e2}) = (g^a, g^b)$.
- **Enc**(m, pk_e): To encrypt message m, randomly choose $r, s \in \mathbb{Z}_q$, compute

$$\tau = (\tau_1, \tau_2, \tau_3) = (pk_{e1}^r, pk_{e2}^s, g^{r+s}m).$$

- **Dec**(τ, sk_e): Given τ and sk_e, compute $m = \frac{\tau_3}{\tau_1^{\frac{1}{a}}\tau_2^{\frac{1}{b}}}$.

Specifically, we denote the rerandomization algorithm by **ReRand**$(p, pk_e, (\tau_1, \tau_2, \tau_3))$, which produces a new ciphertext $(\tau_1(g^a)^{r'}, \tau_2(g^b)^{s'}, \tau_3 g^{r'+s'})$, equivalent to the input ciphertext τ, under the public key $pk_e = (g^a, g^b)$, using the additional random numbers r' and s'.

The ETS functionality

ETS functionality is composed of ETS.Setup, ETS.Sign, ETS.Verify. We give the concrete construction as follows:

- **ETS.Setup**$(params, \lambda, k, n)$:
  1. Parse parameter $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, q)$.
  2. For users(participants), generate public keys and private shares by running (**VK**, *params*, g_1, g_2, u', **U**, **SK**)← Setup$(params, \lambda, k, n)$.
  3. For receiver(verifier), randomly choose $a, b \in \mathbb{Z}_q$ as the receiver's private key sk_e, compute receiver's public key $pk_e = (pk_{e1}, pk_{e2}) = (g^a, g^b)$.
- **ETS.Sign**$(\mathbf{SK}, m, p, pk_e)$: For $m = m_1\, m_2 \cdots m_n \in \{0, 1\}^n$, works as follows:
  1. Randomly choose $r_j \in \mathbb{Z}_q$, and compute σ_j←**Share-Sign**(sk_j, m), that is

$$\sigma_j = (\sigma_{j1}, \sigma_{j2}) = (sk_j(u' \prod_{i=1}^{n} u_i^{m_i})^{r_j}, g^{r_j}).$$

2. Verify the validity of signature σ_j by

$$\hat{e}(\sigma_{j1}, g) = \hat{e}(g_2, vk_j)\hat{e}((u'\prod_{i=1}^{n} u_i^{m_i})^{r_j}, \sigma_{j2}).$$

3. Compute the combined signature $(\overline{\sigma_1}, \overline{\sigma_2}) \leftarrow$ **Combine**$(p, m, j, \sigma_j)_{j\in\Phi}$, that is

$$(\overline{\sigma_1}, \overline{\sigma_2}) = (\prod_{j\in\Phi} \sigma_{j1}^{\lambda_j}, \prod_{j\in\Phi} \sigma_{j2}^{\lambda_j})$$

for each $j \in \Phi$, where a subset $\Phi \subset \{1, 2, \cdots, n\}$ and $|\Phi| = k$. Let λ_j be the Lagrange coefficients so that $\alpha = f(0) = \sum_{j\in\Phi} \lambda_j f(j)$.

4. Randomly choose $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$, encrypt $(\overline{\sigma_1}, \overline{\sigma_2})$ under the receiver's public key $S_1 \leftarrow$ **Enc** $(pk_e, \overline{\sigma_1})$ and $S_2 \leftarrow$ **Enc** $(pk_e, \overline{\sigma_2})$, that is

$$\begin{aligned} S_1 &= (pk_{e1}^{x_1}, pk_{e2}^{x_2}, g^{x_1+x_2}\overline{\sigma_1}) \\ &= (pk_{e1}^{x_1}, pk_{e2}^{x_2}, g^{x_1+x_2}\prod_{j\in\Phi} \sigma_{j1}^{\lambda_j}) \\ &= (pk_{e1}^{x_1}, pk_{e2}^{x_2}, g^{x_1+x_2}\prod_{j\in\Phi} (sk_j(u'\prod_{i=1}^{n} u_i^{m_i})^{r_j})^{\lambda_j}) \end{aligned}$$

$$\begin{aligned} S_2 &= (pk_{e1}^{y_1}, pk_{e2}^{y_2}, g^{y_1+y_2}\overline{\sigma_2}) \\ &= (pk_{e1}^{y_1}, pk_{e2}^{y_2}, g^{y_1+y_2}\prod_{j\in\Phi} (g^{r_j})^{\lambda_j}). \end{aligned}$$

5. Output encrypted threshold siganture (S_1, S_2).

- **ETS.Verify**(p, sk_e, S_1, S_2, m): Parse $p = (\mathbf{VK}, params, g_1, g_2, u', \mathbf{U})$, and $m = m_1\, m_2 \cdots m_n \in \{0, 1\}^n$. Decrypt (S_1, S_2) to get $(\overline{\sigma_1}, \overline{\sigma_2})$, that is

$$\begin{aligned} \overline{\sigma_1} &= \frac{g^{x_1+x_2}\prod_{j\in\Phi} (sk_j(u'\prod_{i=1}^{n} u_i^{m_i})^{r_j})^{\lambda_j}}{pk_{e1}^{\frac{1}{a}x_1} pk_{e2}^{\frac{1}{b}x_2}} \\ &= \prod_{j\in\Phi} (sk_j(u'\prod_{i=1}^{n} u_i^{m_i})^{r_j})^{\lambda_j}. \end{aligned}$$

and

$$\begin{aligned} \overline{\sigma_2} &= \frac{g^{y_1+y_2}\prod_{j\in\Phi} (g^{r_j})^{\lambda_j}}{pk_{e1}^{\frac{1}{a}y_1} pk_{e2}^{\frac{1}{b}y_2}} \\ &= \prod_{j\in\Phi} (g^{r_j})^{\lambda_j}. \end{aligned}$$

then verify the encrypted signature by $\hat{e}(\overline{\sigma_1}, g) = \hat{e}(g_2, g_1)\hat{e}(u'\prod_{i=1}^{n} u_i^{m_i}, \overline{\sigma_2})$, else return 0.

The obfuscation of ETS functionality

From the description of the ETS functionality in above section, we regard a family of circuits $C_{ETS} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ for the ETS functionality, C_λ is a group of circuits $C_{p,\mathbf{SK},pk_e}$. We can draw system parameters (**SK**, pk_e, p) from $C_{p,\mathbf{SK},pk_e}$. Given a circuit $C_{p,\mathbf{SK},pk_e}$, the $\mathrm{Obf}_{\mathrm{ETS}}$ works as follows:

- **Obf** $_{\mathrm{ETS}}(C_{p,\mathbf{SK},pk_e})$:

  1. Extract system parameters (pk_e, **SK**, p).
  2. Parse parameter $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, q)$, **SK** = ($sk_1$, sk_2, $\cdots$, sk_n) and **VK** = (vk_1, vk_2, $\cdots$, vk_n).
  3. For each $j \in \{1, 2, \cdots, n\}$, randomly choose $x_{j1}, x_{j2} \in \mathbb{Z}_q$, encrypt user's private share sk_j to run $(pk_{e1}^{x_{j1}}, pk_{e2}^{x_{j2}}, sk_j') \leftarrow \mathrm{Enc}(pk_e, sk_j)$, $sk_j' = g^{x_{j1}+x_{j2}} sk_j$ is an encrypted form of the original signing key sk_j, then compute $vk_j' = g^{x_{j1}+x_{j2}} vk_j$. Suppose $t = (c_{j1}, c_{j2}, c_{j3}) = (pk_{e1}^{x_{j1}}, pk_{e2}^{x_{j2}}, sk_j')$.
  4. Construct an obfuscated circles $R_{p,pke,t}$ that contains the values (p, pk_e, vk_j', t).

- $R_{p,pke,t}$: The obfuscated circuit can be executed on any untrusted cloud server, and it does the following.

  1. On input security parameter λ, the circuit outputs (pk_e, p).
  2. On input message $m = m_1\, m_2 \cdots m_n \in \{0, 1\}^n$, randomly choose $r_j' \in \mathbb{Z}_q$ to run $\sigma_j' = (\sigma_{j1}', \sigma_{j2}') \leftarrow$ **Share-Sign** (sk_j', m), that is

  $$\sigma_{j1}' = g^{x_{j1}+x_{j2}} sk_j (u' \prod_{i=1}^{n} u_i^{m_i})^{r_j'},$$

  and

  $$\sigma_{j2}' = g^{r_j'}.$$

  3. Verify the validity of signature σ_j' by

  $$\hat{e}(\sigma_{j1}', g) = \hat{e}(g_2, vk_j')\hat{e}(u' \prod_{i=1}^{n} u_i^{m_i}, \sigma_{j2}').$$

  4. Compute the combined signature $(\overline{\sigma_1'}, \overline{\sigma_2'}) \leftarrow$ **Combine** $(p, m, \sigma_j')_{j\in\Phi}$, that is

  $$(\overline{\sigma_1'}, \overline{\sigma_2'}) = (\prod_{j\in\Phi} \sigma_{j1}'^{\lambda_j'}, \prod_{j\in\Phi} \sigma_{j2}'^{\lambda_j'})$$

  for each $j \in \Phi$, where a subset $\Phi \subset \{1, 2, \cdots, n\}$ and $|\Phi| = k$. Let λ_j' be the Lagrange coefficients so that $\alpha = f(0) = \sum_{j\in\Phi} \lambda_j' f(j)$.

  5. Compute $c_1 = \prod_{j\in\Phi} (c_{j1})^{\lambda_j'} = pk_{e1}^{\sum_{j\in\Phi} \lambda_j' x_{j1}}$, and $c_2 = \prod_{j\in\Phi} (c_{j2})^{\lambda_j'} = pk_{e2}^{\sum_{j\in\Phi} \lambda_j' x_{j2}}$.

6. Randomly choose $x'_1, x'_2, y'_1, y'_2 \in \mathbb{Z}_q$, rerandomize the generated signature $\overline{\sigma_1}'$ by running $S_1^* \leftarrow \textbf{ReRand}\ (p, pk_e, (c_1, c_2, \overline{\sigma_1}'))$, that is

$$\begin{aligned} S_1^* &= (S_{11}^*, S_{12}^*, S_{13}^*) \\ &= (c_1 pk_{e1}^{x'_1}, c_2 pk_{e2}^{x'_2}, g^{x'_1+x'_2} \prod_{j\in\Phi} \sigma_{j1}'^{\lambda'_j}) \\ &= (pk_{e1}^{\sum_{j\in\Phi}\lambda'_j x_{j1}+x'_1}, pk_{e2}^{\sum_{j\in\Phi}\lambda'_j x_{j2}+x'_2}, g^{x'_1+x'_2} \prod_{j\in\Phi} (g^{x_{j1}+x_{j2}} sk_j (u' \prod_{i=1}^{n} u_i^{m_i})^{r'_j})^{\lambda'_j}) \\ &= (pk_{e1}^{\sum_{j\in\Phi}\lambda'_j x_{j1}+x'_1}, pk_{e2}^{\sum_{j\in\Phi}\lambda'_j x_{j2}+x'_2}, g^{\sum_{j\in\Phi}\lambda'_j (x_{j2}+x_{j1})+x'_1+x'_2} \prod_{j\in\Phi} (sk_j (u' \prod_{i=1}^{n} u_i^{m_i})^{r'_j})^{\lambda'_j}) \end{aligned}$$

and run $S_2^* \leftarrow \textbf{Enc}\ (pk_e, \overline{\sigma_2}')$, that is

$$\begin{aligned} S_2^* &= (S_{21}^*, S_{22}^*, S_{23}^*) \\ &= (pk_{e1}^{y'_1}, pk_{e2}^{y'_2}, g^{y'_1+y'_2} \prod_{j\in\Phi} \sigma_{j2}'^{\lambda'_j}) \\ &= (pk_{e1}^{y'_1}, pk_{e2}^{y'_2}, g^{y'_1+y'_2} \prod_{j\in\Phi} (g^{r'_j})^{\lambda'_j}). \end{aligned}$$

7. Output (S_1^*, S_2^*).

Besides, the polynomial time property is evident as all the calculation here is valid in polynomial time. It is easily to verify that the obfuscated program by theorem 1.

Theorem 1. *The algorithm $R_{p,pke,t}$ can pass verification.*

Proof 1. *For a valid ciphertext (S_1^*, S_2^*), receiver decrypts (S_1^*, S_2^*), the correctness of $R_{p,pke,t}$ is elaborated as follows*:

$$\begin{aligned} \overline{\sigma_1} &= g^{\sum_{j\in\Phi}\lambda'_j (x_{j2}+x_{j1})+x'_1+x'_2} \prod_{j\in\Phi} (sk_j (u' \prod_{i=1}^{n} u_i^{m_i})^{r'_j})^{\lambda'_j} / pk_{e1}^{\frac{1}{a}\sum_{j\in\Phi}\lambda'_j x_{j1}+x'_1} pk_{e2}^{\frac{1}{b}\sum_{j\in\Phi}\lambda'_j x_{j2}+x'_2} \\ &= g^{\sum_{j\in\Phi}\lambda'_j (x_{j2}+x_{j1})+x'_1+x'_2} \prod_{j\in\Phi} (sk_j (u' \prod_{i=1}^{n} u_i^{m_i})^{r'_j})^{\lambda'_j} / g^{\sum_{j\in\Phi}\lambda'_j x_{j1}+x'_1} g^{\sum_{j\in\Phi}\lambda'_j x_{j2}+x'_2} \\ &= \prod_{j\in\Phi} (sk_j (u' \prod_{i=1}^{n} u_i^{m_i})^{r'_j})^{\lambda'_j}. \end{aligned}$$

$$\begin{aligned} \overline{\sigma_2}' &= g^{y'_1+y'_2} (\prod_{j\in\Phi} g^{r'_j})^{\lambda'_j} / pk_{e1}^{y'_1 \frac{1}{a}} pk_{e2}^{y'_1 \frac{1}{b}} \\ &= g^{y'_1+y'_2} \prod_{j\in\Phi} (g^{r'_j})^{\lambda'_j} / g^{y'_1} g^{y'_2} \\ &= \prod_{j\in\Phi} (g^{r'_j})^{\lambda'_j} \end{aligned}$$

The following equation shows that $R_{p,pke},t$ satisfies correctness:

$$\begin{aligned}
\hat{e}(\overline{\sigma_1}, g) &= \hat{e}(\prod_{j\in\Phi}(sk_j(u'\prod_{i=1}^{n}u_i^{m_i})^{r'_j})^{\lambda'_j}, g) \\
&= \hat{e}(\prod_{j\in\Phi}sk_j^{\lambda'_j}, g)\hat{e}(u'\prod_{i=1}^{n}u_i^{m_i}, \prod_{j\in\Phi}g^{r'_j\lambda'_j}) \\
&= \hat{e}(\prod_{j\in\Phi}(g_2^{f(j)})^{\lambda'_j}, g)\hat{e}(u'\prod_{i=1}^{n}u_i^{m_i}, \prod_{j\in\Phi}g^{r'_j\lambda'_j}) \\
&= \hat{e}(g_2, g^{\sum_{j\in\Phi}f(j)\lambda'_j})\hat{e}(u'\prod_{i=1}^{n}u_i^{m_i}, \overline{\sigma_2}') \\
&= \hat{e}(g_2, g^{\alpha})\hat{e}(u'\prod_{i=1}^{n}u_i^{m_i}, \overline{\sigma_2}') \\
&= \hat{e}(g_2, g_1)\hat{e}(u'\prod_{i=1}^{n}u_i^{m_i}, \overline{\sigma_2}').
\end{aligned}$$

Security properties

In the threshold cryptosystem, we should consider a coalition of k curious but honest users attack against the proposed obfuscator. Therefore, we suppose that an adversary is capable of obtaining the private key shares of corrupted users against the obfuscator, excepting the user who generates the obfuscated implementation as a challenge, that is, an adversary can access the corruption oracle on any corrupted user, but corrupt up to $k-1$ of the n players, the set of oracle restrictions dependent on C is defined as $R(C)$. In this paper, we define $R(C) = \{Corruption, |\Phi| \leq k-1\}$, which can be expressed as $Corruption^{|\Phi|\leq k-1}$. Some security requirements of the proposed obfuscator are introduced in the following descriptions.

Definition 5. [34] *An obfuscator Obf for C meets the ACVBP w.r.t. dependent oracle set $T(C)$ and restricted dependent oracle set $R(C)$ if the following situation holds: There exists a PPT simulator S such that, for distinguisher D, arbitrary polynomial f, all sufficiently large $\lambda \in \mathbb{N}$, and arbitrary $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,*

$$\left|\Pr\left[\begin{array}{l} C\longleftarrow C_\lambda; \\ C'\longleftarrow \mathrm{Obf}(C); \\ b\longleftarrow D^{\ll C,T(C),R(C)\gg}(C',z); \end{array} : b=1\right] - \Pr\left[\begin{array}{l} C\longleftarrow C_\lambda; \\ C''\longleftarrow S^{\ll C\gg}; \\ b\longleftarrow D^{\ll C,T(C),R(C)\gg}(C'',z); \end{array} : b=1\right]\right| < \frac{1}{f(\lambda)},$$

where $D^{\ll C,T(C),R(C)\gg}$ means that D has sampling access to all oracles contained in $T(C)$ and $R(C)$ in addition to C.

Definition 6. *Let (KG, Enc, Dec) and (Setup, Share − Sign, Share − Verify, Combine, Verify) be a couple of linear encryption and the threshold signature algorithms. The threshold algorithm*

is existentially unforgeable w.r.t. ETS functionality if the following situation has to be satisfied: There exists a PPT algorithm A, all sufficiently large $\lambda \in \mathbb{N}$, *arbitrary polynomial f, and arbitrary* $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\left|\Pr\left[\begin{array}{c}(p,\mathbf{SK})\longleftarrow Setup(params,\lambda,k,n),(sk_e,pk_e)\longleftarrow KG(params);\\ (m,\sigma,Q)\longleftarrow A^{\ll Share-Sign_{p,sk_i},Corruption^{|\Phi|\leq k-1}\gg}(p,pk_e,z);\\ 1\longleftarrow Verify(m,\sigma,p),m\notin Q;\end{array}\right]\right|<\frac{1}{f(\lambda)},$$

where $Share-Sign_{p,sk_i}$ *is the Share-Sign oracle,* $Corruption^{|\Phi|\leq k-1}$ *is the corruption oracle such as no more than* $k-1$ *private key shares can be obtained by adversary A in the whole game, Q is the set of message queried by A adaptively.*

Definition 7. *Let (KG, Enc, Dec) and (Setup, Share-Sign, Share-Verify, Combine, Verify) be a couple of linear encryption and the threshold signature algorithms. The threshold signature algorithm is existentially unforgeable w.r.t. the ETS Obfuscator if the following situation has to be satisfied: There exists a PPT algorithm A, all sufficiently large* $\lambda \in \mathbb{N}$, *arbitrary polynomial f, and arbitrary* $z \in \{0,1\}^{\mathrm{poly}(\lambda)}$,

$$\left|\Pr\left[\begin{array}{c}(p,\mathbf{SK})\longleftarrow Setup(params,\lambda,k,n),(sk_e,pk_e)\longleftarrow KG(params);\\ C'\longleftarrow \mathrm{Obf}(C);\\ (m,\sigma,Q)\longleftarrow A^{\ll Share-Sign_{p,sk_i},Corruption^{|\Phi|\leq k-1}\gg}(p,pk_e,z);\\ 1\longleftarrow Verify(m,\sigma,p),m\notin Q;\end{array}\right]\right|<\frac{1}{f(\lambda)},$$

where $Share-Sign_{p,sk_i}$ *is the share sign oracle,* $Corruption^{|\Phi|\leq k-1}$ *is the corruption oracle such as no more than* $k-1$ *private key shares can be obtained by adversary A in the whole game, Q is the set of message queried by A adaptively.*

Correctness

In this section, we identify the following goals that the obfuscator for ETS should satisfy.

1. Correctness: The correctness of an obfuscator requires "Preserving Functionality" as described in Definition 4.
2. Security: The obfuscator needs satisfy ACVBP with respect to $T(C)$ and $R(C)$ and existentially unforgeable with respect to ETS Obfuscator.

Below, we state the Theorem 2 which is a key result used to show the correctness of our construction.

Theorem 2. (Preserving Functionality) *The obfuscated program preserves the functionality of original ETS.*

Proof 2. *On receiving the encrypted threshold signature* (S_1, S_2), *that is*

$$S_1 = (pk_{e1}^{x_1}, pk_{e2}^{x_2}, g^{x_1+x_2}\prod_{j\in\Phi}(sk_j(u'\prod_{i=1}^{n}u_i^{m_i})^{r_j})^{\lambda_j})$$

and

$$S_2 = (pk_{e1}^{y_1}, pk_{e2}^{y_2}, g^{y_1+y_2}\prod_{j\in\Phi}(g^{r_j})^{\lambda_j})$$

where $x_1, x_2, y_1, y_2 \in \mathbb{Z}_q$.

On receiving the obfuscated program (S_1^*, S_2^*), *that is*

$$S_1^* = (pk_{e1}^{\sum_{j\in\Phi}\lambda'_j x_{j1}+x'_1}, pk_{e2}^{\sum_{j\in\Phi}\lambda'_j x_{j2}+x'_2}, g^{\sum_{j\in\Phi}\lambda'_j(x_{j2}+x_{j1})+x'_1+x'_2}\prod_{j\in\Phi}(sk_j(u'\prod_{i=1}^{n}u_i^{m_i})^{r'_j})^{\lambda'_j})$$

and

$$S_2^* = (pk_{e1}^{y'_1}, pk_{e2}^{y'_2}, g^{y'_1+y'_2}\prod_{j\in\Phi}(g^{r'_j})^{\lambda'_j})$$

where $x'_1, x'_2, y'_1, y'_2, r'_j \in \mathbb{Z}_q$.

We observe that both (S_1, S_2) *and* (S_1^*, S_2^*) *are identically distributed.*

Security proof

Theorem 3. *Under the DLLN assumption, the algorithm* $\mathrm{Obf}_{\mathrm{ETS}}$ *is ACVBP with respect to dependent oracle* $T(C) = Share - Sign_{p,sk_i}$ *and restricted dependent oracle* $R(C) = Corruption^{|\Phi|\leq k-1}$.

Proof 3. *Suppose* $C = C_{p,\mathbf{SK},pk_e}$, $T(C) = Share - Sign_{p,sk_i}$ *and* $R(C) = Corruption^{|\Phi|\leq k-1}$. *There are a pair of probabilities* $(\mathrm{Pr}_{Nick}, \mathrm{Pr}_{Junk})$ *that represent* $D^{\ll C,T(C),D(C)\gg}$ *outputs 1, given the true and imitated distributions, respectively. We show that* $\mathbf{S\,K} = (sk_1, sk_2, \cdots, sk_n)$ *and* $\overline{\mathbf{Junk}} = (Junk_1, Junk_2, \cdots, Junk_n)$ *are encrypted in the true and imitated distributions. Since the algorithm* $\mathrm{Obf}_{\mathrm{ETS}}$ *is equivalent to the values* (p, pk_e, vk'_i, t). *So we can utilize a simulator S which imitates these values with sampling access to C. The values* (p, pk_e) can be easily draw from *C*. In order to simulate (t, vk'_i). *Then S chooses n junk values and encrypts them using the receiver's public encryption key* pk_e.

The detailed procedure of S is as below.

1. *Using the sampling access to* $C_{p,\mathbf{SK},pk_e}$ *to get* (p, pk_e).
2. *Parse* $p = (\mathbf{V\,K}, params, g_1, g_2, u', \mathbf{U})$, $\mathbf{V\,K} = (vk_1, vk_2, \cdots, vk_n)$ and $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, q)$.
3. *Randomly choose* $Junk_1, Junk_2, \cdots, Junk_n \in \mathbb{G}$, and $\overline{x_{i1}}, \overline{x_{i2}} \in \mathbb{Z}_q$.
4. *Encrypt* $Junk_i$ *using public key* pk_e, $(c_{i1}, c_{i2}, c_{i3}) = (pk_{e1}^{\overline{x_{i1}}}, pk_{e2}^{\overline{x_{i2}}}, g^{\overline{x_{i1}}+\overline{x_{i2}}}Junk_i)$ *for* $i = 1$ *to* n.
5. *Compute* $\overline{vk_i} = g^{\overline{x_{i1}}+\overline{x_{i2}}}vk_i$ *for* $i = 1$ *to* n.
6. *Set* $Junk = (c_{i1}, c_{i2}, c_{i3})$, *where* $i = 1, 2, \cdots, n$.
7. *Output* $(p, pk_e, \overline{vk_i}, Junk)$, *obviously,* $R_{p,pk_e,Junk}$ *has the same distribution as* $R_{p,pk_e,t}$.

We will first prove that the output distributions of the simulator and the obfuscator are indistinguishable. We prove this by contradiction, assume that the probability that a distinguisher $D^{\ll C,T(C),D(C)\gg}$ *can distinguish between the probabilities described is not negligible. That is,*

$|\Pr_{Nick} - \Pr_{Junk}|$ *is not negligible.*

$$\Pr_{Nick} = \left| \Pr \left[\begin{array}{c} (p, \mathbf{SK}) \longleftarrow \mathrm{Setup}(params, \lambda, k, n), (sk_e, pk_e) \longleftarrow \mathrm{KG}(params); \\ (c_{i1}, c_{i2}, c_{i3}) \longleftarrow \mathrm{Enc}(pk_e, sk_i), i = 1, 2, \cdots, n; \\ sk'_i = c_{i3}, i = 1, 2, \cdots, n; \qquad : b = 1 \\ b \leftarrow D^{\ll C_{p,\mathbf{SK},pk_e}, Share-Sign_{p,sk_i}, Corruption^{|\Phi| \leq k-1} \gg}(p, pk_e, sk'_i, z); \end{array} \right] \right|$$

$$\Pr_{Junk} = \left| \Pr \left[\begin{array}{c} (p, \mathbf{SK}) \longleftarrow \mathrm{Setup}(params, \lambda, k, n), (sk_e, pk_e) \longleftarrow \mathrm{KG}(params); \\ \overline{\mathbf{Junk}} = (Junk_1, Junk_2, \cdots, Junk_n) \in \mathbb{G}^n; \\ (c_{i1}, c_{i2}, c_{i3}) \longleftarrow \mathrm{Enc}(pk_e, Junk_i), i = 1, 2, \cdots, n; \\ sk'_i = c_{i3}, i = 1, 2, \cdots, n; \qquad : b = 1 \\ b \leftarrow D^{\ll C_{p,\mathbf{SK},pk_e}, Share-Sign_{p,sk_i}, Corruption^{|\Phi| \leq k-1} \gg}(p, pk_e, sk'_i, z); \end{array} \right] \right|$$

Assume that the probability of D to win is not negligible, then we build a couple of adversaries (A, B)*, which attacks the semantic security of the encryption algorithm. First, A does as below*:

1. *Take as input* $(params, pk_e, p, z)$.
2. *Parse* $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, q)$.
3. *Generate the signers' private key shares* $\mathbf{SK}$.
4. *Parse* $\mathbf{SK} = (sk_1, sk_2, \cdots, sk_n)$.
5. *Randomly choose* $Junk_1, Junk_2, \cdots, Junk_n \in \mathbb{G}$.
6. *Set* $m_1 = sk_i$ *and* $m_2 = Junk_i$.
7. *Output* (m_1, m_2, pk_e).

Given an encryption ciphertext ct of m_i*, the algorithm B can make a distinction between* m_1 *and* m_2 *by utilizing D.*

1. *Take as input* $(p, pk_e, m_1, m_2, ct, z)$.
2. *Parse* $params = (\mathbb{G}, \mathbb{G}_T, \hat{e}, g, q)$ *and ct.*
3. *Simulate* $D^{\ll C, T(C), D(C) \gg}(p, pk_e, sk'_i, c_{i1}, c_{i2}, z)$.
4. *Output D's output.*

The advantage of attacker B is the same as the advantage of the distinguisher D to distinguish the output distributions of obfuscator and simulator. So if it's not negligible, then it contradicts the DLLN assumption. Thus the advantage of D is negligible when given one tuple of ciphertexts, then the advantage when given three tuples is also negligible. So we conclude that the obfuscator satisfies ACVBP with dependent oracle set $T(C)$ *and restricted oracle set*$R(C)$.

Theorem 4. *If* $\mathrm{Obf}_{\mathrm{ETS}}$ *for ETS functionality is ACVBP w.r.t. dependent oracle* $T(C) = Share - Sign_{p,sk_i}$ *and restricted dependent oracle* $R(C) = Corruption^{|\Phi| \leq k-1}$*, then the existentially unforgeable w.r.t. ETS functionality implies the existentially unforgeable w.r.t. ETS obfuscator.*

Proof 4. *The proof of this theorem is very similar to the proof in* [32]*, see [103, Theorem 1], we thus omit the formal proof here.*

From Theorem 3 and Theorem 4, the TS scheme satisfies the existentially unforgeable, even if the adversary can obtain the obfuscated circuit. The obfuscator for ETS is mainly to enhance the security, and it is safe for the obfuscation circuit to be executed by any untrusted cloud server, and the cloud server could not get any useful information from it.

Corollary 1. *Under DLLN and CDH assumptions, TS scheme is existentially unforgeable w.r.t.* $\mathrm{Obf}_{\mathrm{ETS}}$.

Experimental results

Theoretical performance analysis

Here we analyze the performance efficiency of our scheme, in terms of computational complexity when performing ETS.Sign, $\mathrm{Obf}_{\mathrm{ETS}}$, $R_{p,pke},t$ and ETS.Verify operations. The result is showed in Table 1. In this table, Rand denotes the operation that randomly selects element, Add denotes addition, Mult denotes multiplication, Exp be an exponent operation, Inv denotes inverse operation. As shown in Table 1, the computational complexity of ETS.Sign and $R_{p,pke},t$ algorithms is linear in the number of n and k. All these operations are polynomial bounded operations and can be computed effectively. Therefore, all algorithms are efficient from a theoretical perspective.

Implementation

To provide numerical results, we implement it to measure the performance of our scheme. Our implementation is written in *C* using the Pairing-Based Cryptography Library [40]. For the computations, we use the curve groups that are implemented in the Libpbc library. The computations are run on a PC with 3.70 GHz CPU frequency, and 4 GB of RAM. In the experiment, we use elliptical curves with a base field size of 512 bits and an embedding degree of 2. The security levels are selects as $|p| = 512$.

The following results denote the average running times of related cryptographic operations. In the experiment, the experimental result is the average number of 10 runs. We measure the running time of four algoritms, that is: ETS.Sign, $\mathrm{Obf}_{\mathrm{ETS}}$, $R_{p,pke,t}$ and ETS.Verify. The performing consequence of our scheme is provided in Fig 1 when $n = 5$ and $k = 3$. It is shown that the obfuscated implementation have high efficiency in general, because the algorithm needs perform more exponent operation.

Figs 2 and 3 show the time variety when the number of n and k as variables, respectively. Fig 2 shows the operations time of ETS.Sign, $\mathrm{Obf}_{\mathrm{ETS}}$ and $R_{p,pke,t}$ when k is set as 3 and the number of n is set varies from 5 to 9 increased by an interval of 1. Fig 3 shows the execution time of the three algorithms when n is set as 7 and the number of k is set varies from 3 to 7 increased by an interval of 1. We observe that $R_{p,pke,t}$, ETS.Sign and $\mathrm{Obf}_{\mathrm{ETS}}$'s time cost increases fastly

Table 1. Computational overhead, where n is the number of users, k is the threshold number.

	Operation	ETS.Sign	$\mathrm{Obf}_{\mathrm{ETS}}$	$R_{p,pke,t}$	ETS.Verify
$\mathbb{Z}_q$	Rand	$n + 4$	$2n$	$n + 4$	0
	Add	2	n	$2k$	0
	Inv	0	0	0	4
$\mathbb{G}$	Mult	$2k + n + 1$	$2n$	$4k + n + 1$	0
	Exp	$2k + 2n + 6$	$3n$	$4k + 2n + 6$	4
	Inv	0	0	0	2
$\mathbb{G}_T$	Mult	1	0	1	1
$\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$	Pair	3	0	3	3

https://doi.org/10.1371/journal.pone.0250259.t001

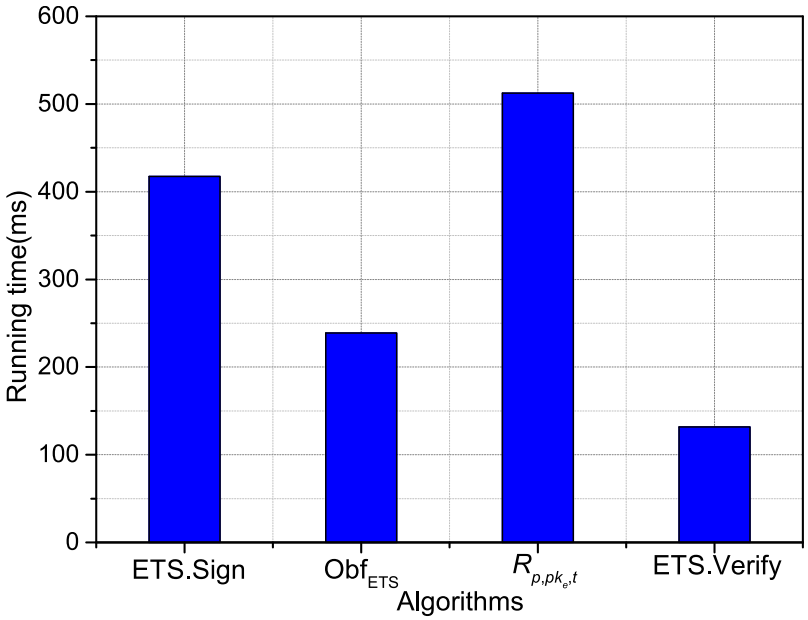

Fig 1. Execution time of the algorithms.

https://doi.org/10.1371/journal.pone.0250259.g001

along with the increasing of n and k. It can be seen from the results that $R_{p,pke,t}$ is more costly than ETS.Sign with the same n or k.

Conclusion

Obfuscation technique can provide much greater security for sensitive data from service providers in cloud computing. In this paper, we design an obfuscator for encrypted threshold signature, according to this technique, key shares are obfuscated before they are uploaded to the

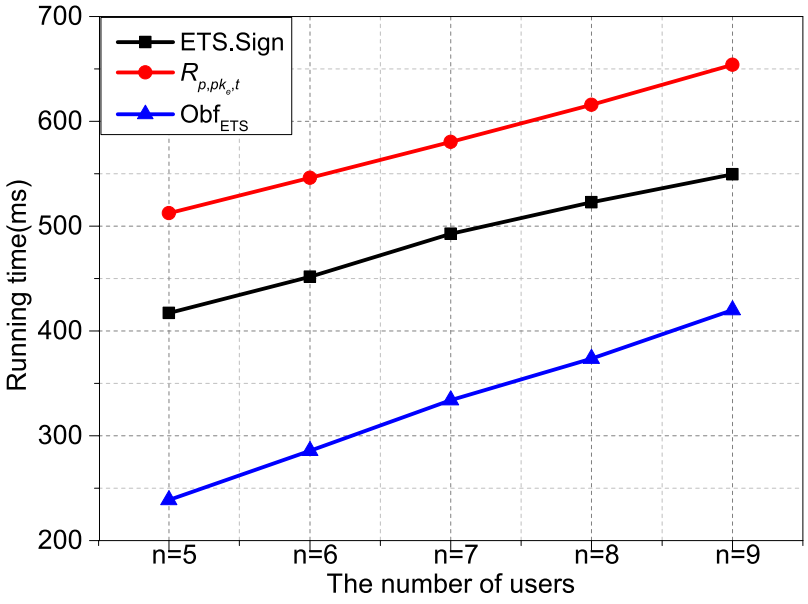

Fig 2. Time cost with k = 3.

https://doi.org/10.1371/journal.pone.0250259.g002

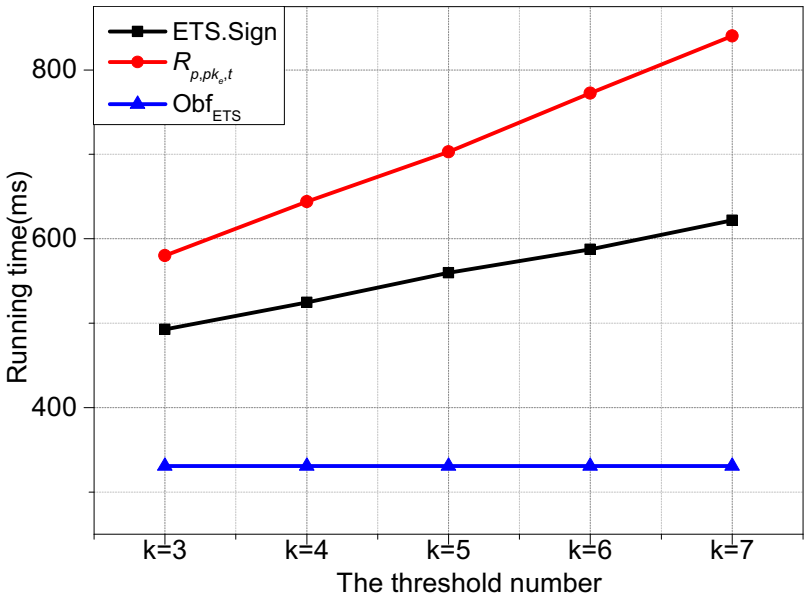

Fig 3. Time cost with n = 7.

https://doi.org/10.1371/journal.pone.0250259.g003

cloud services. In this regard, we can implement the program obfuscator run on a untrusted cloud sever, while hiding privacy-related sensitive information from the obfuscated program. The security analysis demonstrate that our scheme can meet the average case virtual black box property.

Supporting information

S1 Fig. Execution time of the algorithms.
(DOC)

S2 Fig. Time cost with k = 3.
(DOC)

S3 Fig. Time cost with n = 7.
(DOC)

S1 Table. Computational overhead, where n is the number of users, k is the threshold number.
(DOC)

Acknowledgments

The authors would like to thank the anonymous reviewers of this paper for his/her objective comments and helpful suggestions while at the same time helping us to improve the English spelling and grammar throughout the manuscript.

Author Contributions

Conceptualization: Yahong Li, Jianzhou Wei, Bin Wu.

Data curation: Jianzhou Wei, Caifen Wang.

Formal analysis: Chunli Wang, Xiaodong Yang.

Funding acquisition: Xiaodong Yang.

Methodology: Yahong Li, Bin Wu.

Project administration: Xiaodong Yang.

Resources: Yahong Li, Jianzhou Wei.

Software: Chunli Wang, Yulei Zhang.

Supervision: Yahong Li, Caifen Wang.

Validation: Yahong Li.

Writing – original draft: Yahong Li, Jianzhou Wei, Bin Wu.

Writing – review & editing: Yahong Li, Jianzhou Wei, Bin Wu.

References

1. Singh A, Chatterjee K. Cloud security issues and challenges: A survey. Journal of Network and Computer Applications. 2017, 79:88–115. https://doi.org/10.1016/j.jnca.2016.11.027
2. Zhuang W, Ye Q, Lyu F, Cheng N, Ren J. SDN/NFV-Empowered Future IoV With Enhanced Communication, Computing, and Caching. Proceedings of the IEEE, 2019, PP(99):1–18. https://doi.org/10.1109/JPROC.2019.2951169
3. He H, Shan H, Huang A, Ye Q, Zhang W. Edge-Aided Computing and Transmission Scheduling for LTE-U-Enabled IoT. IEEE Transactions on Wireless Communications, to appear. https://doi.org/10.1109/GLOCOM.2018.8647178
4. Ye Q, Li J, Qu K, Zhuang W, Xu L. End-to-End Quality of Service in 5G Networks: Examining the Effectiveness of a Network Slicing Framework. IEEE Vehicular Technology Magazine, 2018, 13(99):65–74. https://doi.org/10.1109/MVT.2018.2809473
5. Zhang Y, Xu C, Li H, Yang K, Zhou J, Lin X. HealthDep: An Efficient and Secure Deduplication Scheme for Cloud-Assisted eHealth Systems. IEEE Transactions on Industrial Informatics, 2018:1–1. https://doi.org/10.1109/TII.2018.2832251
6. Sun P. Security and privacy protection in cloud computing: Discussions and challenges. Journal of Network and Computer Applications, 2020:102642. https://doi.org/10.1016/j.jnca.2020.102642
7. Islam S, Ouedraogo M, Kalloniatis C, Mouratidis H, Gritzalis S. Assurance of Security and Privacy Requirements for Cloud Deployment Model. IEEE Transactions on Cloud Computing, 2018, 6 (99):387–400. https://doi.org/10.1109/TCC.2015.2511719
8. Li J, Zhang Y, Chen X, Xiang Y. Secure attribute-based data sharing for resource-limited users in cloud computing. Comput. Secur. 2018, 72:1–12. https://doi.org/10.1016/j.cose.2017.08.007
9. Mary B, Amalarethinam D. Data Security Enhancement in Public Cloud Storage Using Data Obfuscation and Steganography. IEEE Computer Society. 2017:181–184. https://doi.org/10.1109/WCCCT.2016.52
10. Arul O, Arockiam L. Confidentiality Technique for Enhancing Data Security using Encryption and Obfuscation in Public Cloud Storage. International Journal of Advanced Research in Computer and Communication Engineering. 2016, 9(10): 1–11. https://doi.org/10.17148/IJARCCE.2016.5294
11. Cheng R, Zhang F. Obfuscation for multi-use re-encryption and its application in cloud computing. Concurrency & Computation Practice & Experience. 2015, 27(8):2170–2190. https://doi.org/10.1002/cpe.3399
12. Rehman A, Hussain M. Efficient Cloud Data Confidentiality for DaaS. International Journal of Advanced ence & Technology. 2011, 35: 1–10.
13. Gautam P, Ansari M, Sharma S. Enhanced Security for Electronic Health Care Information Using Obfuscation and RSA Algorithm in Cloud Computing. International Journal of Information Security and Privacy, 2019, 13(1):59–69. https://doi.org/10.4018/IJISP.2019010105
14. Oli SA, Arockiam L. Confidentiality Technique to Encrypt and Obfuscate Non-Numerical and Numerical Data to Enhance Security in Public Cloud Storage. 2017 World Congress on Computing and Communication Technologies (WCCCT). IEEE Computer Society, 2017. https://doi.org/10.1109/WCCCT.2016.51

15. Sugumar R, Joycee K. Ensure and Secure Data Confidentiality in Cloud Computing Environment using Data Obfuscation Technique. International Journal of Advanced Studies in Computer Science and Engineering, 2017, 12(6): 16–21.

16. Zhang Y, Xu C, Lin X, Shen X. Blockchain-Based Public Integrity Verification for Cloud Storage against Procrastinating Auditors. IEEE Transactions on Cloud Computing, 2019:1–1. https://doi.org/10.1109/TCC.2019.2908400

17. Zhang Y, Xu C, Cheng N, Li H, Yang H. Chronos+: An Accurate Blockchain-Based Time-Stamping Scheme for Cloud Storage. IEEE Transactions on Services Computing, 2020, 13(2):216–229. https://doi.org/10.1109/TSC.2019.2947476

18. Zhang Y, Xu C, Ni J, Li H, Shen X. Blockchain-assisted Public-key Encryption with Keyword Search against Keyword Guessing Attacks for Cloud Storage. IEEE Transactions on Cloud Computing, 2019, PP(99):1–1. https://doi.org/10.1109/TCC.2019.2923222

19. Caddy T, Smith S, Stavrou A, Weaver N, Naccache D, Kuhn M, et al. Threshold Cryptography. Encyclopedia of Cryptography and Security. 2015:1288–1293. https://doi.org/10.1007/978-1-4419-5906-5_330

20. Shamir A. How to Share a Secret. Communications of the Acm. 2011, 22(11): 612–613. https://doi.org/10.1145/359168.359176

21. Li J, Yuen T H, Kim K. Practical Threshold Signatures Without Random Oracles. International Conference on Provable Security. Springer, Berlin, Heidelberg, 2007. 10.1007/978-3-540-75670-5_14

22. Pedersen T. A threshold cryptosystem without a trusted party. in: Advances in Cryptology, Proceedings of the Eurocrypt'91, 8-11 April, Brighton, UK, in: LNCS, vol. 547, Springer-Verlag, Berlin, 1991, pp. 522–526. 10.1007/3-540-46416-6_47

23. Gennaro R, Jarecki S, Krawczyk H, Rabin T. Secure Distributed Key Generation for Discrete-Log Based Cryptosystem. In Advances in Cryptology-EUROCRYPT'99, LNCS 1592, Springer-Verlag, pages 295–310, 1999. https://doi.org/10.1007/3-540-48910-X_21

24. Zhang Y, Xu C, Li H, Yang K, Shen X. PROTECT: Efficient Password-based Threshold Single-sign-on Authentication for Mobile Users against Perpetual Leakage. IEEE Transactions on Mobile Computing, 2020, PP(99):1–1. https://doi.org/10.1109/TMC.2020.2975792

25. Saroj S, Chauhan S, Sharma A, Vats S. Threshold Cryptography Based Data Security in Cloud Computing. IEEE International Conference on Computational Intelligence &*amp*; Communication Technology. IEEE, 2015. https://doi.org/10.1109/CICT.2015.149

26. Barak B, Goldreich O. Impagliazzo R, Rudich S, Sahai A, Vadhan S, et al. On the (im)possibility of obfuscating programs. Journal of the ACM. 2010, 59(2):1–6. https://doi.org/10.1145/2160158.2160159

27. Canetti R, Kalai Y, Varia M, Wichs D. On symmetric encryption and point obfuscation. In: TCC'10. Lecture Notes in Computer Science, vol. 5978, pp. 52–71. Springer, Berlin (2010). https://doi.org/10.1007/978-3-642-11799-2_4

28. Hofheinz D, Malone-Lee J, Stam M. Obfuscation for cryptographic purposes. Journal of Cryptology. 2010, 23(1):121–168. https://doi.org/10.1007/s00145-009-9046-1

29. Bellare M, Stepanovs I. Point-function obfuscation: A framework and generic constructions. Theory of Cryptography. TCC 2016. Lecture Notes in Computer Science, vol 9563. Springer, Berlin, Heidelberg. 10.1007/978-3-662-49099-0_21

30. Canetti R, Rothblum G, Varia M. Obfuscation of hyperplane membership. International Conference on Theory of Cryptography. Springer-Verlag, 2010. https://doi.org/10.1007/978-3-642-11799-2_5

31. Hohenberger S, Rothblum G, Shelat A, Vaikuntanathan V. Securely obfuscating re-encryption. Journal of Cryptology. 2011, 24(4), 694–719. https://doi.org/10.1007/s00145-010-9077-7

32. Hada S. Secure Obfuscation for Encrypted Signatures. International Conference on Advances in Cryptology-eurocrypt. Springer-Verlag, 2010. https://doi.org/10.1007/978-3-642-13190-5_5

33. Nishimaki R, Xagawa K. Verifiably encrypted signatures with short keys based on the decisional linear problem and obfuscation for encrypted VES. Designs Codes &*amp*; Crytography. 2015, 77(1):61–98. https://doi.org/10.1007/s10623-014-9986-9

34. Shi Y, Zhao Q, Fan H, Liu Q. Secure obfuscation for encrypted group signatures. PloS One. 2015, 10 (7):1–40. https://doi.org/10.1371/journal.pone.0131550 PMID: 26167686

35. Shi Y, Zhang Q, Liang J, He Z, Fan H. Obfuscatable Anonymous Authentication Scheme for Mobile Crowd Sensing. IEEE Systems Journal, 2018:2918–2929. https://doi.org/10.1109/JSYST.2018.2881711

36. Zhang H, Zhang F, Cheng R, Tian H. Efficient obfuscation for CNF circuits and applications in cloud computing. Soft Computing. 2019, 23: 2061–2072. https://doi.org/10.1007/s00500-017-2921-z

37. Zhang M, Zhang Y, Jiang Y, Shen J. Obfuscating EVES Algorithm and Its Application in Fair Electronic Transactions in Public Clouds. IEEE Systems Journal, 2019:1–9. https://doi.org/10.1109/JSYST.2019.2900723

38. Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: Proc. CRYPTO 2001. LNCS, 2001, 2139:213–229. https://doi.org/10.1007/3-540-44647-8_13

39. Boneh D, Boyen X, Shacham H. Short group signatures. Advancs in Cryptology—Crypto 2004, Proceedings. pp. 41–55. https://doi.org/10.1007/978-3-540-28628-8_3

40. Lynn B, et al. Pairing-based crytography library. 2013, (https://crypto.stanford.edu/pbc/).