Computer Networks 198 (2021) 108374

Contents lists available at ScienceDirect

Computer Networks

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

Group authentication for cloud-to-things computing: Review and improvement

Rui Xu [a,*], Xu Wang [b], Kirill Morozov [c]

[a] School of Computer Science, China University of Geosciences (Wuhan), China
[b] School of Computer Science, University of Science and Technology of China, China
[c] Department Computer Science and Engineering, University of North Texas, United States of America

ARTICLE INFO

Keywords:
Group authentication
Secret sharing
Cryptanalysis
Internet of things
Ad-hoc networks

ABSTRACT

Group authentication enables a set of users to mutually authenticate each other without the help of a coordinating manager, which has proven effective for the new framework of cloud-to-things computing. Recently many group authentication schemes have been proposed using threshold secret sharing technique. In this article we review these schemes and show that some of group authentication schemes based on secret sharing are insecure. Specifically we develop an impersonation attack against such schemes. This attack allows an outsider with no credential to successfully authenticate herself to a group of users. The idea behind our attack is quite simple: in the threshold setting, when the size of the group is larger than the threshold an outsider can compute a linear combination of honest users' tokens which validates her identity though she is actually not a registered member in the group. Finally we propose three types of security enhancement to fix the vulnerability. Our improved group authentication schemes avoid the above attack by breaking the linearity of threshold secret sharing, using the technique of signature, splitting communication and commitment scheme.

1. Introduction

Continuous development of the Internet of Things (IoT) is fueling the Big Data growth. According to a forecast [1] from International Data Corporation (IDC), the number of total connected IoT devices, or "things", will reach 41.6 billion, generating 79.4 zettabytes (ZB) of data in 2025. As the devices connected into IoT have been increasing in numbers, new paradigm of cloud-to-things computing [2] emerges to connect IoT with the mature technology of cloud computing so as to realize the great potential of IoT. Further development of cloud-to-things computing framework includes fog computing [3] and edge computing [4]. Fog computing aims at processing data near the source in order to reduce network traffic as opposed to relying on a central cloud server. Edge computing also tends to process the data at the "edge" of the network (i.e., closer to the end users) to save bandwidth cost and to address the concerns of response time requirement, battery life constraints, as well as data security and privacy. Fig. 1 shows a typical framework of edge computing which combines edge nodes, fog nodes and a cloud server.

As illustrated in Fig. 1, there are three layers in the edge computing framework. The edge nodes at the bottom represent various devices connected into the IoT. The edge nodes may consist of remote sensors, smart meters, and wearable devices which are usually resource-constrained. The fog nodes connect the cloud server and the edge nodes. The fog nodes can be considered more powerful than edge nodes and they are responsible for managing a group of edge nodes.

Security and privacy threats [5,6] are a major concern for the wide deployment of IoT applications since IoT devices normally collect physical measurements (such as electricity consumption) and personal information (such as health data) which are sensitive. In this paper we consider the problem of authentication. In IoT applications, especially in the edge computing framework, communication occurs in groups where a group of edge devices under a certain fog node exchange information about their collected data and working status. In some applications, the number of devices within a group can be as large as thousands. Traditional one-to-one authentication solutions thus cannot be employed in the IoT situation due to the natural characteristic of limited resource in IoT devices.

Harn [7] proposed group authentication as a measure to enable many-to-many authentication in group communication. In group authentication schemes (GAS), a set of group members can mutually authenticate each other effectively without relying on authenticating in a pairwise manner or sending messages to a central server. We briefly introduce the setting of group authentication scheme as follows. A set of users form a group and there is a group manager (or group leader)

* Corresponding author.
E-mail address: ruixu.cug@qq.com (R. Xu).

https://doi.org/10.1016/j.comnet.2021.108374
Received 9 August 2020; Received in revised form 19 July 2021; Accepted 27 July 2021
Available online 2 August 2021

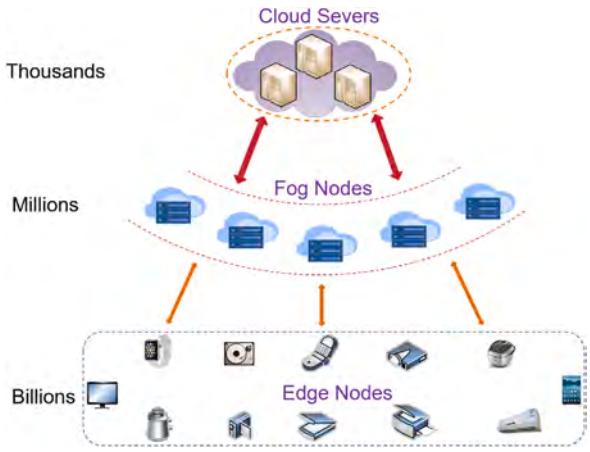


Fig. 1. Edge computing framework.

who is more powerful than other users. The communication network in the group can be synchronous or asynchronous. The group manager is responsible for distributing initial private keys to each member in the group. When a subset of users in the group want to initiate a group session they use their private keys to start a group authentication protocol. At the end of the GAS, if all users are legitimate, they can mutually authenticate each other. We provide an informal description of the layout of (t, m, n) GAS below.

In a (t, m, n) GAS, there are in total n users forming a communication group and , in addition, a group manager. When a set of m ($m \geq t$) users launch a group authentication, if all the m users are legitimate then they can mutually authenticate each other in the sense that they all belong to the same group of size n. Furthermore, if at most $t-1$ out of the m insider users are dishonest, any outside attacker can neither impersonate a legitimate user nor get the private key of any other honest insider user.

Based on Shamir's secret sharing scheme, Harn proposed three GAS schemes. Their first GAS works in synchronous communication network, which is not very practical. Their second GAS works in asynchronous network but it is only one-time secure, meaning that the private keys can be used just once. Their third GAS is secure in asynchronous network and works for multiple authentications. However, Chien [8] pointed out a weakness of Harn's asynchronous GAS for multiple authentications that if an attacker joins the group authentication for multiple trials then she can deduce the private keys of legitimate users and thus launch impersonation attack. In order to fix Harn's asynchronous GAS, Chien proposed a group authentication scheme using Shamir secret sharing, elliptic curve cryptography (ECC) and pairing-based cryptography. Chien's scheme is claimed to be secure for multiple authentications and multiple trials.

Our Contributions

1. We find that Chien's GAS is not secure under impersonation attack. In this work, we show an impersonation attack against Chien's scheme even under one-time authentication. In addition, we point out that some other group authentication schemes based on secret sharing have the same vulnerability as that of Chien's GAS.
2. In order to enhance the security of group authentication schemes based on secret sharing, we propose three methods to improve the GAS. Our proposed group authentication schemes are secure under impersonation attack and enable multiple authentications and multiple trials.

2. Preliminaries

In this section we introduce some necessary background.

2.1. Shamir secret sharing

Shamir [9] proposed a (t, n) threshold secret sharing scheme to split a secret s into n pieces s_i called shares. The n shares are distributed to n different users $(U_1, \ldots, U_n)$. The basic properties of Shamir's (t, n) threshold secret sharing scheme are: 1) The collection of less than t shares reveals no information of the secret s; 2) The collection of t or more shares uniquely determines the original secret s. We briefly introduce Shamir's construction as follows.

For a secret $s \in \mathbb{F}_p$, the dealer chooses a random polynomial $f(x) \in \mathbb{F}_p[x]$ of degree at most $t-1$ over the polynomial ring $\mathbb{F}_p[x]$. The free coefficient of $f(x)$ is the secret s, i.e., $f(0) = s$. We can write down $f(x)$ as $f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$, where $a_i \in \mathbb{F}_p$, for $i = 1, 2, \ldots, t-1$, is chosen uniformly at random. Each user U_i is identified by a unique and distinct element $x_i \in \mathbb{F}_p$. The share for user U_i is the evaluation of $f(x)$ on her identifying element x_i as $s_i = f(x_i)$. Due to Lagrange interpolation, t or more users can pull their shares together to interpolate the polynomial $f(x)$ and hence they can reconstruct the shared secret s. At the same time, less than t shares reveal no information about the secret s. When m ($m \geq t$) users $\mathbf{U}_m = \{U_1, \ldots, U_m\}$ want to reconstruct the secret s, each of them computes a reconstruction component $c_i = s_i \cdot l_{i,\mathbf{U}_m}$. The secret is then computed as $s = \sum_{i=1}^{m} c_i$. Here, $l_{i,\mathbf{U}_m} = \prod_{j=1, j\neq i}^{m} \frac{x_j}{x_j - x_i}$ is called Lagrange coefficient and can be calculated from public information, specifically the identifying elements $(x_1, \ldots, x_m)$ of the m users $\mathbf{U}_m$. Since the Lagrange coefficient $l_{i,\mathbf{U}_m}$ depends on the particular users in the reconstruction, we associate it with a subscript $\mathbf{U}_m$.

2.2. Bilinear map

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order q, where $\mathbb{G}_1$, $\mathbb{G}_2$ are additive groups and $\mathbb{G}_T$ is a multiplicative group. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a pairing which satisfies the following properties:

- Bilinear: $e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$ and $e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2)$.
- Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_2$ to the identity in $\mathbb{G}_T$.
- Computability: There exists an efficient algorithm to compute $e(P, Q)$ for all $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$.

The Weil or Tate pairings associated with supersingular elliptic curves satisfy the above properties. An elliptic curve $E(\mathbb{F}_p)$ over finite field $\mathbb{F}_p$ is the set of points $P = (x, y)$, for $x, y \in \mathbb{F}_p$, satisfying the equation $y^2 = x^3 + ax + b \in \mathbb{F}_p[x]$.

Next we define some problems related to elliptic curves and bilinear pairings. Those problems are widely believed to be intractable for properly selected parameters.

Definition 1. Elliptic Curve Discrete Logarithm Problem (ECDLP): given an elliptic curve $E(\mathbb{F}_p)$ and two points $P, Q \in E(\mathbb{F}_p)$, to find an integer k such that $Q = kP$.

Definition 2. Computational Elliptic Curve Diffie–Hellman Problem (ECDHP): given an elliptic curve $E(\mathbb{F}_p)$, a point $P \in E(\mathbb{F}_p)$, and two points $A = aP$, $B = bP$, to find a point $C = abP$.

Definition 3. Bilinear Pairing Inversion Problem (BPIP): given a bilinear pairing e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ as described above, $e(P, Q) \in \mathbb{G}_T$ and $Q \in \mathbb{G}_2$, to find $P \in \mathbb{G}_1$.

2.3. Cryptographic primitives

In this subsection we briefly introduce some useful cryptographic primitives.

A cryptographic hash function H: $D \to R$ maps an element $x \in D$ to its hash digest $h = H(x) \in R$. The domain D can be of arbitrary size, while the range R if of fixed size. One of their commonly required properties is collision resistance. It states that it should be difficult to find two different inputs x_1 and x_2 such that $H(x_1) = H(x_2)$.

A cryptographic hash function is sometimes modeled as a random oracle [10], which is a function sampled uniformly from the set of all functions from D to R. When the security proof relies on modeling hash functions as random oracles, we say that the underlying protocol is secure in the random oracle model.

A commitment scheme is a cryptographic primitive that allows one to "commit" to a chosen message m while keeping it hidden till the opening is made [11, Chapter 5.6.5]. Informally, it satisfies the following two properties.

1. Hiding. The commitment value reveals no information about m.
2. Binding. It is difficult for the committer to open a commitment to two different messages m and m'.

A commitment scheme can be easily constructed using a random oracle H [11] as follows. The commitment to a message m is $Com = H(r|m)$, where r is a (large enough) value chosen uniformly at random. When the message m is itself random, the value r can be omitted.

3. Review of group authentication schemes

In this section we review some of group authentication schemes based on secret sharing. The initial work of Harn [7] used Shamir secret sharing as a building block. Chien [8] later pointed out that Harn's asynchronous (t, m, n) GAS for multiple authentications is vulnerable to impersonation attack when the attacker joined the GAS instances from several authentication trials. Ahmadian and Jamshidpour [12] recently proposed a linear subspace cryptanalysis method to Harn's asynchronous GAS. Their attack method allows an outsider to impersonate a group member even if the GAS is used just once. Besides pointing out the vulnerability of Harn's scheme, Chien also proposed a new (t, m, n) GAS which is claimed to be secure against impersonation attack for multiple trials and multiple authentications. However, we point out that Chien's GAS is also vulnerable to impersonation attack. Our attack method is inspired by that of Ahmadian and Jamshidpour, although it is different from theirs. Specifically, the attacking method of Ahmadian and Jamshidpour requires the attacker to observe messages of at least $t + k - 1$ authenticated group members, where k is an additional parameter of Harn's GAS. Our method works when the attacker observes messages of at least t authenticated group members. Actually we find that some other group authentication schemes [13,14], which are based on secret sharing, are also vulnerable to our attacks.

3.1. Chien's group authentication scheme

We find out that some of GAS schemes [8,13,14] based on secret sharing scheme are vulnerable to impersonation attack. Since all these constructions are built in the same spirit, we only consider Chien's scheme in details. *However, we stress that our proposed attacking method extends in a straightforward manner to other schemes [13,14].* Table 1 defines the notations used.

Table 1
Notations.

Notation	Description
GM	Group manager
U_i	The ith user
x_i	Identifying element of user U_i
H	Hash function modeled as random oracle
$\mathbf{U}_m$	A collection of m users
q	A prime number
n	Number of users in a group
t	Threshold of the secret sharing scheme
s	A secret value chosen by the GM to be shared
s_i	Secret share for user U_i
$l_{i,\mathbf{U}_m}$	Lagrange coefficient of user U_i associated with $\mathbf{U}_m$

Initialization. The group manager GM chooses three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order q, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups over elliptic curves and $\mathbb{G}_T$ is multiplicative. Then the GM selects a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The GM further sets a polynomial $f(x) \in \mathbb{F}_q[x]$ with $f(0) = s$, where $s \in \mathbb{F}_q$ is a secret value. For every user U_i in the group $\mathbf{U} = \{U_1, \ldots, U_n\}$, the GM sets an identifying element $x_i \in \mathbb{F}_q$ to her. Each user U_i receives the Shamir secret share $s_i = f(x_i)$ as her private key. Finally the GM chooses an element (an elliptic curve point) $P \in \mathbb{G}_2$ and computes $Q = sP \in \mathbb{G}_2$.[1] The points P and Q are public.

Group Authentication. When a group of m $(m \geq t)$ users $\mathbf{U}_m = \{U_1, \ldots, U_m\}$, with identifying elements $(x_1, \ldots, x_m)$, want to authenticate each other, they proceed as follows.

- The group of m users first agree on some random element $R \in \mathbb{G}_1$.[2]
- Each user $U_i \in \mathbb{U}_m$ computes $c_i = s_i \cdot l_{i,\mathbb{U}_m}$ and $c_i R$.
- Each user releases $c_i R$.
- After all users release their $c_i R$, they can compute $\sum_{i=1}^{m} c_i R$ and check whether the equation $e(\sum_{i=1}^{m} c_i R, P) = e(R, Q)$ holds or not. If the equation holds, then all users accept the protocol. Otherwise, the group authentication fails.

3.2. Impersonation attack on Chien's GAS

Based on hardness assumptions of ECDLP, ECDHP and BPIP, Chien proved the security of his GAS in asynchronous network with multiple authentications and multiple trials. However, we show that Chien's GAS is vulnerable to an impersonation attack of a single outsider even in the one-time setting.

The main claim of security by Chien [8] is that the only condition when the equation $e(\sum_{i=1}^{m} c_i R, P) = e(R, Q)$ holds is that all users in $\mathbf{U}_m$ are authentic users. However, we show that even a single outside user can pretend to be an authentic user and provide transcript which validates the equation $e(\sum_{i=1}^{m} c_i R, P) = e(R, Q)$.

Before presenting our impersonation attack, we list the security description as Claim 1, which is adapted from Lemmas 6–9 in Chien's [8] paper.

Claim 1. *Given a random point $R \in \mathbb{G}_1$ and a group of members $\mathbf{U}_m = \{U_1, \ldots, U_m\}$ with $m \geq t$, the only condition that the group $\mathbf{U}_m$ can reconstruct the values satisfying $e(\sum_{i=1}^{m} c_i R, P) = e(R, Q)$ in Chien's GAS is that all the participating members are valid and the scheme can resist up to $t - 1$ colluded insiders.*

Now we describe our impersonation attack against Chien's GAS. We assume that there are m $(m \geq t)$ honest users $\mathbf{U}_m = \{U_1, \ldots, U_m\}$ and our attacker wants to impersonate a legitimate user claiming identity U_{m+1} with identifying element x_{m+1}. Denote the $m + 1$ users as $\mathbf{U}_{m+1} =$

[1] Note that the group operation in $\mathbb{G}_2$ is addition and hence in the multiplication $Q = sP$, s should be treated as an integer. We can assume that there is a bijective mapping $\sigma : \mathbb{F}_q \to \{0, \ldots, q-1\}$ and $Q = \sigma(s)P$. However, the scheme is designed such that $s \in \mathbb{F}_q$ where q is the order of the cyclic group $\mathbb{G}_2$. Thus we do not explicitly transfer an element of $\mathbb{F}_q$ to integer.

[2] The random element R can be generated using a hash function with the session ID as input.

$\{U_1, \ldots, U_{m+1}\}$. At the group authentication phase, the $m+1$ users agree on a random point $R \in \mathbb{G}_1$. Each of the m honest users, take for example the user U_i, will compute $c_i = s_i \cdot l_{i,\mathbf{U}_{m+1}}$ and $c_i R$. However, the impersonation attacker U_{m+1} is unable to compute c_{m+1} or $c_{m+1}R$ because she does not know the corresponding share s_{m+1}. Nonetheless, we observe that the attacker impersonating U_{m+1} can, however, deduce a valid $c_{m+1}R$ so that $\sum_{i=1}^{m+1} c_i R = sR$ and hence satisfying the equation $e(\sum_{i=1}^{m+1} c_i R, P) = e(R, Q)$.

Recall that $c_i = s_i \cdot l_{i,\mathbf{U}_{m+1}}$ for $i = 1, \ldots, m$. Suppose for a moment that only the m honest users $\mathbf{U}_m$ want to join the group authentication phase, then each user would have computed $c_i' = s_i \cdot l_{i,\mathbf{U}_m}$. Then we have $\sum_{i=1}^{m} c_i' = s$. Further we have

$$c_i' = \frac{c_i \cdot l_{i,\mathbf{U}_m}}{l_{i,\mathbf{U}_{m+1}}} = \frac{x_{m+1} - x_i}{x_{m+1}} c_i \tag{1}$$

for $i = 1, \ldots, m$.

Now the impersonating attacker wants to deduce a c_{m+1} such that it satisfies

$$\sum_{i=1}^{m+1} c_i = s = \sum_{i=1}^{m} c_i'. \tag{2}$$

Substituting Eq. (1) into Eq. (2), we get:

$$\sum_{i=1}^{m} c_i + c_{m+1} = \sum_{i=1}^{m} \frac{x_{m+1} - x_i}{x_{m+1}} c_i. \tag{3}$$

At this point, we can calculate c_{m+1} as:

$$c_{m+1} = \sum_{i=1}^{m} \left(\frac{x_{m+1} - x_i}{x_{m+1}} - 1 \right) c_i = \sum_{i=1}^{m} \frac{-x_i}{x_{m+1}} c_i. \tag{4}$$

At the group authentication phase each user releases just $c_i R$ instead of c_i. By the hardness assumption of ECDLP, our attacker cannot deduce c_i from $c_i R$. However, for the attacker to succeed, she does not necessarily have to learn c_{m+1}. She merely needs to come up with $c_{m+1}R$ which satisfies $\sum_{i=1}^{m+1} c_i R = sR$ and hence $e(\sum_{i=1}^{m+1} c_i R, P) = e(R, Q)$. This is easy since:

$$c_{m+1} R = \sum_{i=1}^{m} \frac{-x_i}{x_{m+1}} c_i R. \tag{5}$$

From Eq. (5), we observe that $c_i R$ for $i = 1, \ldots, m$ are released by honest users and x_i for $i = 1, \ldots, m+1$ are public knowledge. Hence, the impersonating attacker can easily compute a valid $c_{m+1}R$ which authenticates her to the group of users $\mathbf{U}_{m+1} = \{U_1, \ldots, U_{m+1}\}$.

Clearly, our proposed scheme directly disproves Claim 1.

4. Improvement of Chien's GAS

In this section we propose some methods to improve Chien's GAS so that it can resist the impersonation attack we described. As we review the impersonation attack in the previous section, we can observe that the reason that an outsider succeeds in impersonating a user is that she can calculate a linear combination of tokens of honest users. So our proposed improvements all try to address this vulnerability by breaking the linearity.

4.1. Dividing communication into two rounds

Notice that the attacker can compute a valid value $c_{m+1}R$ because it is a linear combination of honest user's $c_i R$ and in an asynchronous network the attacker might see all of these calculated parameters. One method to avoid such attack is to not let the attacker see the released values of the honest users. We propose to divide the communication into two rounds. In the first round, every user just submits a commitment of $c_i R$. Each user then reveals her committed value $c_i R$ in the second round. The commitment of $c_i R$ can be simply made as $H(c_i R)$ assuming that the hash function H is a random oracle. We describe the improved GAS next (see Fig. 2).

Initialization. The initialization phase is almost the same as that of Chien's GAS except that the GM chooses a public collision-resistant hash function H.

Group Authentication. When a group of m users $\mathbf{U}_m = \{U_1, \ldots, U_m\}$ want to authenticate each other, they agree on a random point $R \in \mathbb{G}_1$ and do the following in two rounds.

Round 1. Each user U_i computes $c_i = s_i l_{i,\mathbf{U}_m}$, $c_i R$ and $h_i = H(c_i R)$. In the first round each user U_i releases h_i.

Round 2. Each user U_i releases $c_i R$.

At the end of the second round, every user checks the following equations:

$$h_i = H(c_i R) \text{ for } i = 1, \ldots, m; \tag{6}$$

$$e(\sum_{i=1}^{m} c_i R, P) = e(R, Q). \tag{7}$$

If all these checks pass, the group of m users authenticate each other. Otherwise the group authentication fails.

It is easy to see that an impersonation attacker can no longer just compute a $c_{m+1}R$. In order to successfully impersonate a legitimate user she has to also provide the hash digest $h_{m+1} = H(c_{m+1}R)$ beforehand. If the attacker was not identified by the verification equations Eqs. (6) and (7), then she must have been able to find a collision of the hash function H or have been able to find the pre-image of the hash function H. However, this is hard when H is modeled as a random oracle. We sketch the proof in the following.

Consider an impersonation attacker who claims to be U_{m+1} and gathers together with $\mathbf{U}_m = \{U_1, \ldots, U_m\}$ to launch a group authentication session. Then, according to our revised scheme, the attacker observes $h_i = H(c_i R)$ for $i = 1, \ldots, m$ in the first round and $c_i R$ for $i = 1, \ldots, m$ in the second round. Note that when the group of users $\mathbf{U}_{m+1} = \{U_1, \ldots, U_m, U_{m+1}\}$ and the random element R are fixed, there exists a unique value of $c_{m+1}R$ satisfying Eq. (7). According to Eq. (5), $c_{m+1}R$ is determined by the values of $c_i R$ for $i = 1, \ldots, m$. However, due to perfect secrecy of Shamir threshold scheme, $c_i R$ for $i = 1, \ldots, m$ are perfectly hidden to the attacker since $c_i R$ are computed from the secret share s_i. Unless the attacker finds the pre-image of $h_i = H(c_i R)$, she cannot compute $c_{m+1}R$ by Eq. (5). In another case, the attacker might compute $c_{m+1}R$ by Eq. (5) in the second round after seeing all $c_i R$ for $i = 1, \ldots, m$. However, she has to commit the value $c_i R$ by submitting $h_{m+1} = H(c_{m+1}R)$ in the first round. This means the attacker can open the committed value to a different message, which breaks the binding property of commitment scheme, indicating that the attacker can find a collision of the hash function H. Since we model the hash function H as a random oracle, neither finding the pre-image nor collision of H is possible.

4.2. Signing the message

Note that even though the attacker impersonating user U_{m+1} does not have the secret key s_{m+1}, she can compute a valid value $c_{m+1}R$ from the transcript of the authentication phase. Then our second method to prevent impersonation attack is to ask each user to sign her message in order to prove that she is indeed a legitimate user (has the private key). We describe the second improvement in the following Fig. 3.

Initialization. The initialization phase is similar to that of Chien's GAS. In addition, the GM selects a hash function $H : \mathbb{F}_q \to \mathbb{G}_2$.

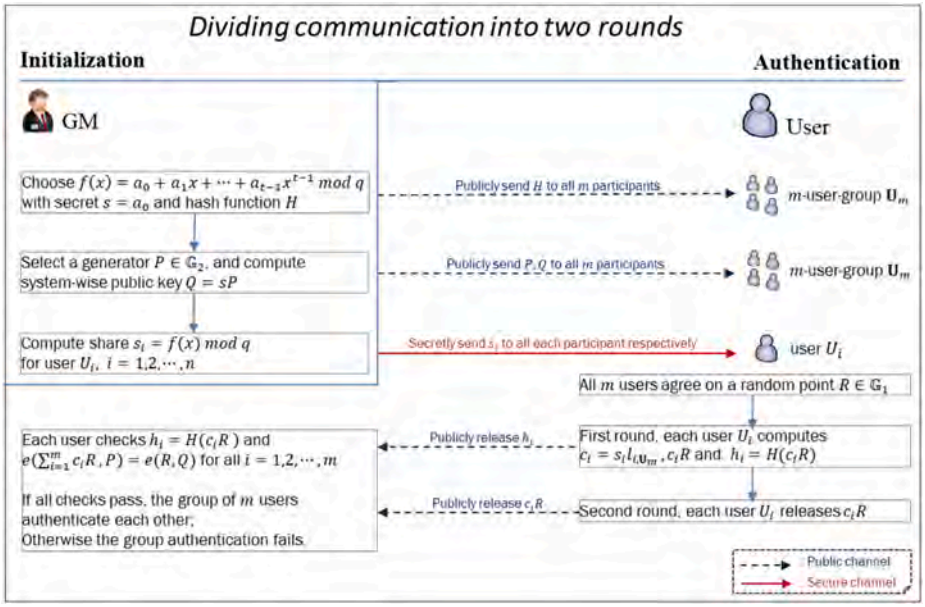


Fig. 2. Dividing communication into two rounds.

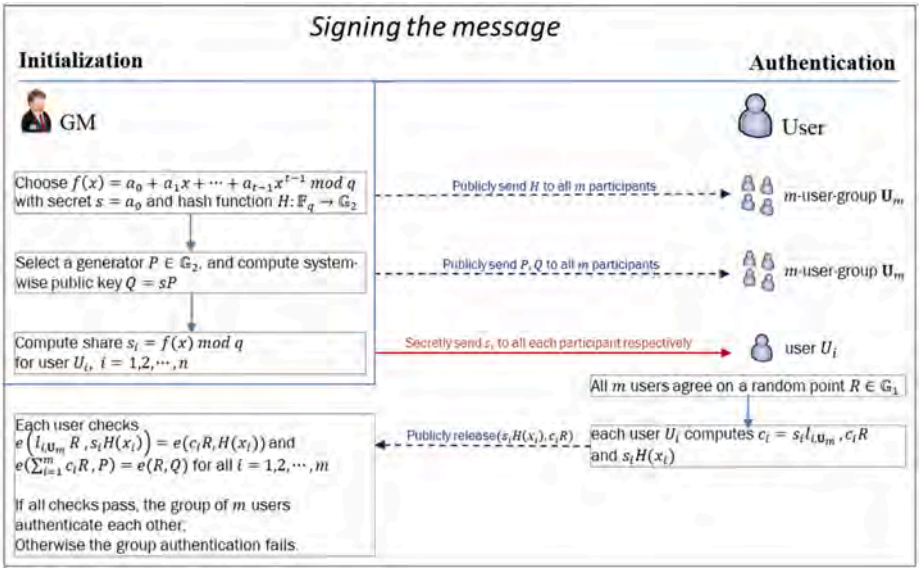


Fig. 3. Signing the messag.

Group Authentication. When a group of m users $\mathbf{U}_m = \{U_1, \ldots, U_m\}$ want to authenticate each other, they agree on a random point $R \in \mathbb{G}_1$. Each user U_i computes $s_iH(x_i)$, $c_i = s_i l_{i,\mathbf{U}_m}$ and c_iR. Then each user U_i releases $(s_iH(x_i), c_iR)$. After all users release their message, each of them checks the following equations:

$$e(l_{i,\mathbf{U}_m}R, s_iH(x_i)) = e(c_iR, H(x_i)) \text{ for } i = 1, \ldots, m; \tag{8}$$

$$e(\sum_{i=1}^{m} c_iR, P) = e(R, Q). \tag{9}$$

If all these equations hold, the group of m users authenticate each other. Otherwise, the group authentication fails.

Note that Eq. (9) is the same as the verification equation of Chien's GAS. But now Eq. (8) can be used to verify whether the user U_i has the private key s_i. If U_i has the private key s_i then we have $e(l_{i,\mathbf{U}_m}R, s_iH(x_i)) = e(s_i \cdot l_{i,\mathbf{U}_m}R, H(x_i)) = e(c_iR, H(x_i))$ by the bilinear property of e. The values $l_{i,\mathbf{U}_m}$ and $H(x_i)$ can be calculated from public information. At the same time, an attacker who does not know s_i is unable to provide $(s_iH(x_i), c_iR)$ which will pass the verification of Eq. (8). The argument is that in Eq. (8) $(c_iR, s_iH(x_i))$ can be treated as a signature of $l_{i,\mathbf{U}_m}R$ using the private key s_i, while the public key of U_i is $H(x_i)$. We stress that the linear combination trick in our impersonation attack does not work for an adversary who wants to forge the signature because the second input of the bilinear map e depends on each user's identifying element. This dependence breaks the linear relationship which was explored in the impersonation attack.

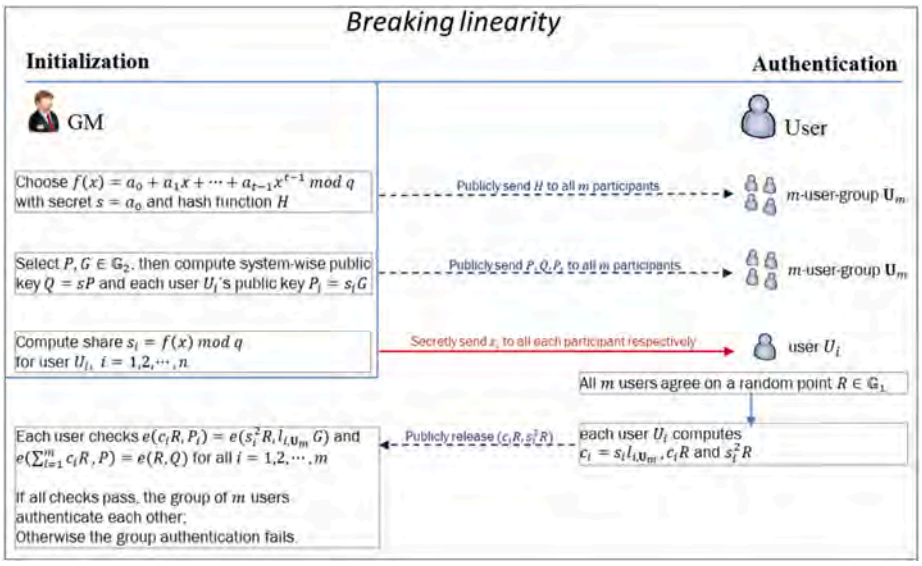


Fig. 4. Breaking linearity.

4.3. Breaking linearity

Since the impersonation attack we proposed uses the linearity between the honest users' message and that of the attacker, we propose another improvement of Chien's GAS by breaking this linearity. We describe the improved GAS next (Fig. 4).

Initialization. The initialization phase is similar to that of Chien's GAS. In addition, the GM selects a point $G \in \mathbb{G}_2$ and computes for each user U_i a public key $P_i = s_iG$. The point G is public. The public keys P_i for $i = 1, \ldots, n$ can be made publicly available. Alternatively, they can be sent to each user and later each user announces their respective public key in the group authentication phase.

Group Authentication. When a group of m users $\mathbf{U}_m = \{U_1, \ldots, U_m\}$ want to authenticate each other, they agree on a random point $R \in \mathbb{G}_1$. Each user U_i computes $c_i = s_i l_{i,\mathbf{U}_m}$, c_iR, and s_i^2R. Then each user U_i releases (c_iR, s_i^2R). After all users release their messages, each of them checks the following equations:

$$e(c_iR, P_i) = e(s_i^2R, l_{i,\mathbf{U}_m}G) \text{ for } i = 1, \ldots, m; \tag{10}$$

$$e(\sum_{i=1}^{m} c_iR, P) = e(R, Q). \tag{11}$$

If all these equations hold, the group of m users authenticate each other. Otherwise, the group authentication fails.

It is easy to verify that if a user U_i indeed has the private key s_i, she can pass the verification equation Eq. (10) since:

$$e(c_iR, P_i) = e(s_i l_{i,\mathbf{U}_m}R, s_iG) = e(s_i^2 l_{i,\mathbf{U}_m}R, G) = e(s_i^2R, l_{i,\mathbf{U}_m}G). \tag{12}$$

But if a user U_i does not have a valid private key s_i, she cannot pass Eq. (10) because she is unable to provide s_i^2R do to the assumed hardness of BPIP.

5. Conclusions

In this paper we review group authentication schemes using the technique of Shamir secret sharing. We identified a vulnerability in some of these schemes and proposed an impersonation attack against the representative one by Chien. Our impersonation attack allows an outsider attacker to successfully impersonate any group member easily. In order for the attacker to succeed, she just needs to compute a linear combination of the transmitted messages sent by honest users.

We also proposed three improved group authentication schemes to fix this vulnerability. Our first improvement requires two rounds of communication and a collision-resistant hash function. Our second improvement requires an ideal hash function which is secure in the random oracle model. Our third improvement is secure in the standard model and does not need further assumption compared with that of Chien's GAS. Another merit of our enhanced schemes is that they can identify dishonest users in the group authentication phase. It is easy to observe that for a certain user U_i, if Eq. (7) , Eq. (8), or Eq. (10) does not verify then she can be identified as dishonest.

Funding

This research was funded by National Natural Science Foundation of China with grant number NSFC 61802354.

CRediT authorship contribution statement

Rui Xu: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing, Funding acquisition. **Xu Wang:** Methodology, Validation, Writing – review & editing, Visualization. **Kirill Morozov:** Conceptualization, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the anonymous reviewers for their useful suggestions and insightful comments during the review process.

Appendix. Our impersonation attack

```
def share(s, n, t, q):
    print('The secret is ', s)
 Zq = Zmod(q)
 s = Zq(s)
 R = Zq['x']
 VS = VectorSpace(Zq, t)
 coeff_vec = VS.random_element()
 coeff_vec[0] = s
 coeff = list(coeff_vec)
 f = R(coeff)
 share_list = [(Zq(xi), f(Zq(xi))) for xi in range(1, n + 1)]
 print('The polynomial is ', f)
 print(share_list)
 return share_list

def get_comp_ecc(sharelist, n, t, q, p):
    userlist = [xi for xi, si in sharelist]
 shadowlist = [si for xi, si in sharelist]
 rec_component = []
 for i in range(len(userlist)):
    xi = userlist[i]
     coeff_lagrange = 1
 for xr in userlist:
 if xr != xi:
 coeff_lagrange *= xr / (xr - xi)
  coeff_lagrange *= shadowlist[i]
  rec_component.append(coeff_lagrange)
 F.<b> = GF(p)
 E = EllipticCurve(F, [0,0,1,1,1])
 R = E(b^26 + b^25 + b^24 + b^22 + b^20 + b^17 + b^16 + b^15 + b^13
     + b^11 + b^8 + b^7 + b^6 + b^5 + b^3 + b^2 + b , b^27 + b^25
     + b^22 + b^21 + b^20 + b^19 + b^18 + b^16 + b^15 + b^14 + b^13
     + b^11 + b^6 + b^3 + b^2 + 1)
 print('The order of R is ', R.order())
 comp_ecc = [ZZ(ci) * R for ci in rec_component]
 return comp_ecc

def forge_ecc(comp_ecc, userlist, n, t, q, p):
 F.<b> = GF(p)
 E = EllipticCurve(F, [0,0,1,1,1])
 R = E(b^26 + b^25 + b^24 + b^22 + b^20 + b^17 + b^16 + b^15 + b^13
     + b^11 + b^8 + b^7 + b^6 + b^5 + b^3 + b^2 + b , b^27 + b^25
     + b^22 + b^21 + b^20 + b^19 + b^18 + b^16 + b^15 + b^14 + b^13
     + b^11 + b^6 + b^3 + b^2 + 1 )
 P = E(b^27 + b^26 + b^25 + b^23 + b^22 + b^18 + b^15 + b^13 + b^12
     + b^7 + b^6 + b^3 + 1, b^25 + b^24 + b^22 + b^19 + b^16 + b^14
     + b^13 + b^12 + b^7 + b^4 + b^2 + 1)
 Q = 99 * P
 forge_component = E(0, 1, 0)
 last_user = userlist[-1]
 userlist = userlist[:-1]
 for ci, xi in zip(comp_ecc, userlist):
 forge_component += ZZ((last_user / (last_user - xi))^(-1) *
 (xi / (xi - last_user))) * ci
 print('Tpye of forge_component is ', type(forge_component))
 forge_ecc_point = sum(comp_ecc) + forge_component
 pairing1 = forge_ecc_point.weil_pairing(P, q)
 pairing2 = R.weil_pairing(Q, q)
 print('Left hand side of verification equation is ', pairing1)
 print('Right hand side of verification equation is ', pairing2)

def main():
```

```
    n = 10
    t = 5
    s = 99
    q = 113
    p = 2**28
    share_list = share(s, n, t, q)
    sharelist = share_list[:t+1]
    print('length of sharelist is ', len(sharelist))
    userlist = [xi for xi, si in sharelist]
    rec_comp_ecc = get_comp_ecc(sharelist, n, t, q, p)
    rec_comp_ecc_without_last = rec_comp_ecc[:-1]
    print('length of rec_comp_ecc_without_last is ',
    len(rec_comp_ecc_without_last))
    forge_ecc(rec_comp_ecc_without_last, userlist, n, t, q, p)

main()
```

References

[1] C. MacGillivray, D. Reinsel, Worldwide global datasphere IoT device and data forecast, 2019–2023, idc market forecast - doc # us45066919, 2019, Google Patents. https://www.idc.com/getdoc.jsp?containerId=US45066919.

[2] S.M. Babu, A.J. Lakshmi, B.T. Rao, A study on cloud based internet of things: CloudIoT, in: 2015 Global Conference on Communication Technologies (GCCT), IEEE, 2015, pp. 60–65.

[3] M. Aazam, E.-N. Huh, Fog computing: The cloud-iot\/ioe middleware paradigm, IEEE Potentials 35 (3) (2016) 40–44.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.

[5] Z.-K. Zhang, M.C.Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, S. Shieh, IoT Security: ongoing challenges and research opportunities, in: 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, IEEE, 2014, pp. 230–234.

[6] M.A. Khan, K. Salah, IoT Security: Review, blockchain solutions, and open challenges, Future Gener. Comput. Syst. 82 (2018) 395–411.

[7] L. Harn, Group authentication, IEEE Trans. Comput. 62 (9) (2012) 1893–1898.

[8] H.-Y. Chien, Group authentication with multiple trials and multiple authentications, Secur. Commun. Netw. 2017 (2017) 1–7, http://dx.doi.org/10.1155/2017/3109624, https://www.hindawi.com/journals/scn/2017/3109624/.

[9] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.

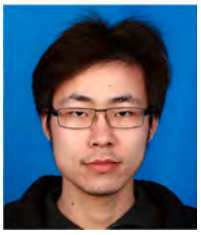
[10] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, in: Proceedings of the 1st ACM Conference on Computer and Communications Security, 1993, pp. 62–73.

[11] J. Katz, Y. Lindell, Introduction To Modern Cryptography, 2nd Ed., CRC press, 2015.

[12] Z. Ahmadian, S. Jamshidpour, Linear subspace cryptanalysis of harn's secret sharing-based group authentication scheme, IEEE Trans. Inf. Forensics Secur. 13 (2) (2017) 502–510.

[13] D.-H. Lee, I.-Y. Lee, Dynamic group authentication and key exchange scheme based on threshold secret sharing for iot smart metering environments, Sensors 18 (10) (2018) 3534.

[14] Y. Aydin, G.K. Kurt, E. Ozdemir, H. Yanikomeroglu, A flexible and lightweight group authentication scheme, IEEE Internet Things J. 7 (10) (2020) 10277–10287.



Dr. Rui Xu received his B.S. and M.S. degree in engineering from University of Science and Technology of China. He received his Ph.D. degree in Mathematics from Kyushu University, in 2015. He worked as an associate researcher in the information security group of KDDI Research Inc., Japan from 2015 to 2018. He is now with University of Geosciences (Wuhan), China. His research areas are secret sharing schemes and cloud security. He is also interested in multi-party computation and privacy preserving techniques.

Xu Wang received her bachelor's degree from China University of Geosciences (Wuhan) in 2019. She is now a master student at the University of Science and Technology of China. Her main research interests are secret sharing and coding. She has obtained two patents on secret sharing in China.

Dr. Kirill Morozov received his M.Sc. degree with Honors in Radiophysics and Electronics from Saint-Petersburg State University of Telecommunications, Russia, in 1998. He received his Ph.D. degree in Computer Science from University of Aarhus, Denmark, in 2005. He has been working in various positions in research and academia for over 15 years, including a postdoc at the University of Tokyo from 2005, a research scientist at the National Institute of Advanced Industrial Science and Technology (AIST), Japan, from 2006, an assistant professor at Kyushu University from 2010, and an adjunct associate professor in Tokyo Institute of Technology from 2016. Since November 2017, he is with the Department of Computer Science and Engineering, University of North Texas, USA, as an associate professor. His research interests include cryptography and cybersecurity, in particular, post-quantum cryptosystems and secret sharing.