cryptography

MDPI

Article

Security and Performance of Single Sign-On Based on One-Time Pad Algorithm

Maki Kihara *,†,‡ **and Satoshi Iriyama** ‡

Department of Information Science, Tokyo University of Science, Yamazaki 2641, Noda, Chiba 278-8510, Japan; iriyama@is.noda.tus.ac.jp

* Correspondence: kiharamaki18@gmail.com

† Research Fellow of Japan Society for the Promotion of Science.

‡ These authors contributed equally to this work.

Received: 13 April 2020; Accepted: 9 June 2020; Published: 12 June 2020

Abstract: Single sign-on (SSO) techniques allow access control for multiple systems with a single login. The aim of our study is to construct an authentication algorithm that provides the authentication information of a user to a requester without requiring any specific token, thereby achieving domain-free access control. In this study, we propose an authentication algorithm for SSO based on a verifiable encryption (VE)-based authentication algorithm and implementation. VE is a kind of cryptosystem that allows calculation on cyphertexts, generating an encrypted result, which matches the distance between two plaintexts when decrypting. In our approach, we first construct the mathematical SSO algorithm based on the VE-based algorithm, and then implement the algorithm by applying the one-time pad to the algorithm and using sample data. We also consider robustness against theoretical attacks such as man-in-the-middle attack. In addition to that, our algorithm is robust against the well-known classical and theoretical attacks, the man-in-the-middle attack against the proposed algorithm is also impracticable. Furthermore, with security analysis using Proverif, the algorithm has been shown to be secure. The execution speed is less than 1 ms even with a text length of 8192 bits. Based on our results, it is evident that the computational burden of trusted third parties, such as a certificate authority, can be alleviated because the public key agreement is not required in our algorithm. Moreover, since only the authentication information is disclosed to the service provider, big tech such as GAFA cannot obtain personal information of the user without consent. As for the originality of our algorithm, any personal information, such as biometric information and non-contact magnetic IC cards in addition to the pair of ID and password, which is used for common SSO algorithms, is available.

Keywords: single sign-on; authentication; one-time pad; cryptography; security

1. Introduction

According to the 2019 mid-year estimates of the world internet usage and population statistics, there are more than 4.5 billion internet users in the world, and this number is persistently increasing [1]. From 2000 to 2019, the number of internet users increased by 1157%; this significant increase in the number of users can also be attributed to the remarkable developments in network technologies.

In general, a system or device accepts or rejects a user's request for a network service by first verifying the user's identity when it receives the request. In a traditional electronic system or device, the validity of a service request is verified by checking whether the pair of ID and password provided by the user matches the stored ID and password information. This identity verification process is called authentication, which refers to the act of checking whether the identity provided by Alice is the same as the identity of Alice held by Bob. Here, Bob is called a verifier, while Alice is called a

prover. Many network systems verify a user's request via authentication and then determine whether to provide the requested service to the user based on authorization. Thus, authorization is a process to determine whether a requested action is allowed, and this is often performed after authentication for many network services.

Internet users often access multiple network services, and are required to be authenticated for each network service. However, the users do not often create different pairs of IDs and passwords for different services because it is cumbersome to manage multiple pairs of information; therefore, the same ID and password pair tends to be reused several times. Such reuse of personal authentication information can be abused by malicious third parties. Hence, to prevent the personal information of users from being compromised, service providers must implement extensive identity control. However, considering the high costs involved in the security management of user identity information, if possible, service providers should avoid storing such information.

In recent years, a technique called single sign-on (SSO) that enables access control for multiple systems by allowing one-time access to a specific system has been proposed as a suitable solution to the above-mentioned problems [2]. According to [3], SSO of the market is more than 800 million US dollars in the world at the time of 2016, and it is estimated to grow to about 160 billion US dollars by 2021. SSO is indispensable for improving the productivity in IT environments where cloud computing is progressing.

According to [4], there are the following five benefits of using these SSO technologies.

1. Reduces Help Desk costs;
2. Improves customer satisfaction;
3. Boosts productivity;
4. Improves compliance and security capabilities;
5. Facilitates B2B collaboration.

As a first benefit, users do not need to assign and remember passwords for a plurality of systems and services with SSO. Reference [4] states that up to 50% of the inquiries of the service provider help desk are only password resets. To counter threats such as unauthorized access to data and personal information, some companies have requested the use of complex and long passwords as one of their strong security policies. Recent users access multiple (not a few) applications and are required to enter the ID and a long and complex password many times per day to log in. Namely, users are required to remember a large number of long and complex passwords. SSO helps to reduce the chance that users forget their password and consequently decreases the temporal and financial costs associated with password-related issues, such as account blocking due to forgotten passwords or incorrect entries, and password assistance. The next benefit is improving customer satisfaction. According to [4], many sites providing SSO such as social networking services value user experience and have a user-friendly login process. SSO is laid out to improve the user experience when logging in by a quick and simple process. Moreover, SSO reduces the number of long and complex passwords entered during login and the number of help desk requests if the users make a mistake, thus reducing the user wait-time. As a result, SSO boosts productivity, which is the third benefit. Businesses encourage employees to use security applications such as a secure file transfer system to protect sensitive data. In fact, these applications are underutilized because users usually find them too complicated. Against this problem, since SSO can reduce the hassle of logging in, it can help encourage employees to adhere to security policies of the organizations. Moreover, SSO increases the chance of avoidance of password reuse and the use of easily guessable passwords. Hence, SSO helps improve compliance and security capabilities. Facilitation of B2B collaboration is listed as the last benefit. Using SSO technologies, the companies can centralize and simplify the management of authentication and authorization. Moreover, users have to login once and then simply and quickly access any participating partners' shared applications.

According to [5–7], there are four primary architectures for Web SSO; these are listed in Table 1. In the token-based SSO approach, tokens and tickets are distributed between the interacting parties if user authentication based on user credentials containing an ID and password pair is successful. There

are two types of token-based SSO implementations, namely the Kerberos authentication protocol and SSO for cookies. In the former case, an authentication ticket is transported using a method called remote procedure call (RPC), which allows a program to execute the procedures and subroutines present in another address space. Please refer to [8,9] for more details. In contrast, in the server-side credential caching approach, all credentials are stored in a central repository; however, a cache is also maintained on the server side. There are several SSO implementations depending on the network environment and requirements.

Table 1. Single sign-on (SSO) architectures and corresponding examples.

Types	Example
Token-based SSO	Kerberos
Public Key Infrastructure (PKI)-based SSO	BrowserID
Client-side credential caching	Windows Credential Manager
Server-side credential caching	CA ETrust SSO

As examples of common used SSO implementation, there are Kerberos [8,9], security assertion markup language (SAML) [10], and OpenID [11]. Not only Kerberos, SAML and OpenID but also OAuth [12] is one of the well-known SSO technologies. However, OAuth is not focused in this paper because OAuth is a technology that focuses on authorization rather than authentication. Kerberos is an authentication method in which the user is authenticated only with the ID and password for the first time and thereafter utilizes the identity certificate called a ticket. In Kerberos authentication, if the user sends the correct ID and password and then succeeds in authentication, the user receives a ticket. The server of the service provider determines whether to allow or reject the user by checking whether the user has a ticket. SAML is an XML-based markup language for security assertions and provides three types of statements: authentication, attribute, and authorization decision. In SAML, the user ID of the identity provider and service provider need to correspond to each other before the SSO process where the identity provider is a trusted entity that issues or registers a user's identity. Thus, SAML requires building up trustful relations between the identity provider and service provider. OpenID is an authentication method that allows the use of common ID information on various websites. It provides only authentication information in contrast with SAML, which handles authentication, authorization, and attribute information. A server, which is provided by a company called Big Tech such as GAFA, serves as the identity provider and service provider and provides authentication to other service providers. Here, Big Tech refers to the largest and most paramount companies in the information technology industry. Moreover, OpenID differs from SAML in that there is no pre-established trust relationship between the identity provider and service provider, and it does not need security tokens such as assertions.

However, according to the survey by [13], a certain number of people refuse or hesitate to use WebSSO. One of the reasons why network users refuse to use WebSSO may be because they are hesitant to disclose their personal information. In fact, [14] says that the government and the companies collect personal information to monitor personal lives without permission of individuals is regarded as a problem. For example, personal information of users who use SSO (search history, purchase history, etc.) may be disclosed to SSO providers and used for marketing, or posted advertisements on web browsers without user permission. Such issues are called the consent management issues. Since many SSO mechanisms require the user to provide an ID and password directly, it is regarded problematic in [6] because many people also have the impression that sensitive personal information is stored somewhere locally using the mechanism. Although various SSO products currently exist, unified rules and verification regarding security have not been achieved. (See [15–18].) Therefore, the SSO algorithm in which personal information can be used only for authentication as originally intended, and enables to maintain security is required.

The aim of this study is to realize construction of an SSO algorithm that provides only authentication information without disclosing the user identity and sensitive personal information to the service provider. We propose an authentication algorithm based on verifiable encryption, which is a type of cryptosystem that allows calculation on the space of cyphertexts and returns an encrypted result representing the distance between two plaintexts. The SSO algorithm is constructed by applying a one-time pad, and the VE-based authentication algorithm proposed in the previous paper [19]. Both the VE-based authentication algorithm and the algorithm proposed in this paper do not need to store any personal information in local storage. Moreover, many user identities such as biometric information and unique numbers without limiting the usable personal information of the ID can be applied to not only VE-based authentication algorithms but also to the algorithm proposed in this paper.

The structure of this paper is outlined as follows: in Section 2, we describe our methodology. In Section 3, we define a cryptosystem and verifiable encryption and present the VE-based authentication algorithm, and then construct the SSO algorithm based on a VE-based authentication algorithm. In addition, the theorem for a mathematical subclass of VE and the theorem associated with the algorithm proposed in this study are proven in this section. In Section 4, the demonstration of our SSO algorithm is shown. We discuss (1) the robustness of the theoretical attacks against our algorithm; (2) the comparison with those of the above-mentioned major SSO implementations, viz. Kerberos, SAML, and OpenID; and (3) the impact on customer and business in Sections 5 and 6 concludes the paper.

2. Methodology

As the background of this research, the current SSO technology has issues of consent management and security as mentioned in the introduction. The purpose of this research is to construct an SSO algorithm that solves these issues. That is, **our purpose** is to construct the SSO algorithm that achieves

- Maintenance of security
- Without unauthorized disclosure of user's personal information

Especially, we aim for the construction of the algorithm in which the user's personal information cannot be leaked in principle, and assume that the service provider cannot obtain anything other than authentication information.

To complete the purpose, we try to achieve the objective by constructing an algorithm that satisfies the following conditions as **our research strategy**.

- The affiliation of each information such as the user's personal information, the key for encryption the user's one, and the personal information of the service provider is clarified.
- The user's personal information and the key used to encrypt the user's one are not stored in the same storage.
- The user's personal information and authentication information are not possessed at the same time.
- The key used to encrypt user's personal information and authentication information are not possessed at the same time.

In **our research design**, the VE-based authentication algorithm and its implementation are utilized for the construction SSO algorithm. The VE-based authentication realizes secure and fast authentication for unlocking a local device via a network without key distribution to the server. The server never knows user's personal information and its authentication information. Moreover, not only ID and password but also arbitrary personal information, for example, biometric information and unique numbers held by individuals such as credit card numbers and contactless magnetic IC card numbers can be applied to the VE-based algorithm.

Our research approach is as follows:

- Construction to mathematical SSO algorithm based on VE-based authentication algorithm;
- Prototype implementation and testing;
- Security discussion.

Our research procedure used is the following:

1. Define parties;
2. Define network configuration;
3. Define parameters such as plaintext, etc.;
4. Define the affiliation of each parameter;
5. Construct mathematical algorithm;
6. Implement.

3. Algorithm

SSO access control is a property of software systems that allows a user to log in to some approved systems using a single login account. In this section, we introduce the authentication procedure based on VE proposed in [19]; in addition, we describe our modifications to this algorithm to realize SSO.

3.1. Cryptosystem and Authentication Algorithm Based on VE

Let $\mathcal{P}, \mathcal{C}$, and $\mathcal{K}$ be spaces of plaintexts, cyphertexts, and keys, respectively. Then, sets $\mathcal{E}$ and $\mathcal{D}$ of encryptions and decryptions, respectively, can be given by

$$\mathcal{E} = \{E_k : \mathcal{P} \to \mathcal{C} | E_k(p) = c, \forall p \in \mathcal{P}, k \in \mathcal{K}\},$$
$$\mathcal{D} = \{D_k : \mathcal{C} \to \mathcal{P} | D_k(c) = p, \forall c \in \mathcal{C}, k \in \mathcal{K}\}.$$

Definition 1. *A 5-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is called a cryptosystem if*

$$\forall p \in \mathcal{P}, \exists k \in \mathcal{K}, D_k(E_k(p)) = p.$$

Let $V : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+ (= [0, +\infty))$ be a metric between two texts.

Definition 2. *For a given metric V and two cryptosystems $C_1 = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and $C_2 = (\mathcal{P}, \mathcal{C}, \mathcal{K}', \mathcal{E}', \mathcal{D}')$, a set $(\mathcal{E}, \mathcal{E}')$ is called a VE, if for all $p_1, p_2 \in \mathcal{P}$, there exist maps $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and $D : \mathcal{C} \to \mathbb{R}_+$, and two keys $(k, k') \in \mathcal{K} \times \mathcal{K}'$ such that*

$$D_{k,k'}(F(E_k(p_1), E'_{k'}(p_2))) = V(p_1, p_2),$$

where $E \in \mathcal{E}$ and $E' \in \mathcal{E}'$.

Here, we introduce the proposed authentication algorithm based on VE, which includes a registration step to enroll a user's secret information for a system or service, and a verification step to verify this information. As a prerequisite, we describe the entities and channels used here as follows.

- S is a computation server and untrusted party;
- Alice is a prover;
- Bob is a verifier;
- The channel between Alice and Bob is secure;
- The channel between Bob and S is insecure;
- Alice does not have direct access to S.

Let $C_1 = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and $C_2 = (\mathcal{P}, \mathcal{C}, \mathcal{K}', \mathcal{E}', \mathcal{D}')$ be two cryptosystems where the set $(\mathcal{E}, \mathcal{E}')$ is a VE. Let $p_1, p_2 \in \mathcal{P}$ be two plaintexts, $(k, k') \in \mathcal{K} \times \mathcal{K}'$ be two keys, and

$$c_1 = E_k(p_1) \in \mathcal{C},$$
$$c_2 = E'_{k'}(p_2) \in \mathcal{C}$$

be two cyphertexts.

Registration step

Step1 Alice sends p_1 to Bob.
Step2 Bob generates k and calculates $E_k(p_1) = c_1$.
Step3 Bob sends c_1 to the server S.

Verification step

Step1 Alice sends p_2 to Bob.
Step2 Bob generates k' and calculates $E'_{k'}(p_2) = c_2$.
Step3 Bob sends c_2 to S.
Step4 Server S calculates $F(c_1, c_2) = c_d$, and sends c_d to Bob.
Step5 Bob calculates $D_{k,k'}(F(c_1, c_2))$, and checks the result.

In this algorithm, the key distribution between Alice and Bob or Bob and the server is not required. The distance between two plaintexts can be essentially calculated on the cyphertexts space on the server without decryption. Furthermore, the distance between two plaintexts can be obtained by applying the map $D_{k,k'}$ to $F(c_1, c_2)$ where k and k' are two keys. It is noteworthy that when a one-time pad is applied as a cryptosystem, minimal computational resources are required [19].

3.1.1. Mathematical Subclass of VE

Let $(G, \circ)$ be an abelian group. We construct two maps F and D using the operator $\circ$, and discuss the mathematical subclass of VE.

Theorem 1. *If a cryptosystem $C = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, a metric V and two maps F and D satisfy the following conditions, this cryptosystem belongs to the class of VE.*

1. $\mathcal{P}, \mathcal{C}$, and $\mathcal{K}$ are the abelian groups that belong to a group G closed by the operator $\circ$
2. $\mathcal{E} = \{E_k | E_k(p) = p \circ k, \forall p \in \mathcal{P}, k \in \mathcal{K}\}$
3. $\mathcal{D} = \{D_k | D_k(c) = c \circ k^{-1}, \forall p \in \mathcal{P}, k \in \mathcal{K}\}$
4. Under the operation $\circ$, $V : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$ is composed as $V(p_1, p_2) := p_1 \circ (p_2)^{-1} (\forall p_1, p_2 \in \mathcal{P})$
5. Under the operation $\circ$, $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ is composed as $F(c_1, c_2) := c_1 \circ (c_2)^{-1} (\forall c_1, c_2 \in \mathcal{C})$
6. Under the operation $\circ$, $D : \mathcal{C} \to \mathbb{R}_+$ is composed as $D_{k,k'}(c) := c \circ (k \circ (k')^{-1})^{-1} (\forall c \in \mathcal{C})$

Proof. Let $C = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a cryptosystem such as

- $\mathcal{P}, \mathcal{C}$, and $\mathcal{K}$ are the same arbitrary abelian groups, and $\mathcal{P}, \mathcal{C}, \mathcal{K} \subset G$, where G is a group closed by the operator $\circ$
- $\mathcal{E} = \{E_k | E_k(p) = p \circ k, \forall p \in \mathcal{P}, k \in \mathcal{K}\}$
- $\mathcal{D} = \{D_k | D_k(c) = c \circ k^{-1}, \forall p \in \mathcal{P}, k \in \mathcal{K}\}$

Let V be a metric as $V(p_1, p_2) = p_1 \circ {p_2}^{-1}$. Let $C_1 = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ and $C_2 = (\mathcal{P}, \mathcal{C}, \mathcal{K}', \mathcal{E}', \mathcal{D}')$ be two cryptosystems that are the same as cryptosystem C. Let $p_1, p_2 \in \mathcal{P}$ be two plaintexts, $(k, k') \in \mathcal{K} \times \mathcal{K}'$ be two keys, and

$$c_1 = p_1 \circ k \in \mathcal{C},$$
$$c_2 = p_2 \circ k' \in \mathcal{C}.$$

be the corresponding cyphertexts. Here, we construct maps $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and $D : \mathcal{C} \to \mathbb{R}_+$ as

$$\begin{aligned} F(E_k(p_1), E'_{k'}(p_2)) &:= E_k(p_1) \circ (E'_{k'}(p_2))^{-1} \\ D_{k,k'}(c) &:= c \circ (k \circ (k')^{-1})^{-1}, \end{aligned}$$

respectively. Furthermore, we calculate

$$\begin{aligned} D_{k,k'}(F(c_1, c_2)) &= F(c_1, c_2) \circ (k \circ (k')^{-1})^{-1} \\ &= (E_k(p_1) \circ (E'_{k'}(p_2))^{-1}) \circ (k'^{-1})^{-1} \circ k^{-1} \\ &= (p_1 \circ k) \circ (p_2 \circ k')^{-1} \circ k' \circ k^{-1} \\ &= p_1 \circ k \circ (k')^{-1} \circ (p_2)^{-1} \circ k' \circ k^{-1} \\ &= (p_1 \circ (p_2)^{-1}) \circ (k \circ k^{-1}) \circ (k'^{-1} \circ k') \\ &= p_1 \circ (p_2)^{-1} \\ &= V(p_1, p_2). \end{aligned}$$

For all $p_1, p_2 \in \mathcal{P}$, there exist keys k, k', such that the metric $V(p_1, p_2)$ can be achieved. Therefore, this cryptosystem that satisfies the above-mentioned conditions belongs to the class of VE. □

Corollary 1. *The cryptosystem composed of a Caesar cypher belongs to the class of VE.*

Proof. The cryptosystem of a Caesar cypher can be represented as $C_{cs} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where

$$\begin{aligned} &\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26} = \{0, 1, \cdots 25\}, \\ &\mathcal{E} = \{E_k | E_k(p) = p + k \mod 26, \forall p \in \mathcal{P}, k \in \mathcal{K}\}, \\ &\mathcal{D} = \{D_k | D_k(c) = c - k \mod 26, \forall c \in \mathcal{C}, k \in \mathcal{K}\}. \end{aligned}$$

Here, $a - b \mod 26$ can be considered as $a + (-b) \mod 26$, where $(-b)$ is the inverse of b. $\mathbb{Z}_{26}$ is an abelian group closed by addition $+$. Hence, C_{cs} satisfies conditions 1, 2, and 3 in Theorem 1.

Let $p_1, p_2 \in \mathcal{P}$ be two plaintexts, $(k, k') \in \mathcal{K} \times \mathcal{K}'$ be two keys, and

$$\begin{aligned} c_1 &= p_1 + k \in \mathcal{C}, \\ c_2 &= p_2 + k' \in \mathcal{C}. \end{aligned}$$

be the corresponding cyphertexts. We define a metric $V : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_+$ as $V(p_1, p_2) = |p_1 - p_2|$. Because $-p_2$ is the inverse of p_2, V can be considered as being composed in the addition operation $+$. Furthermore, we construct two maps $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and $D : \mathcal{C} \to \mathbb{R}_+$ as follows:

$$\begin{aligned} F(c_1, c_2) &:= c_1 - c_2 \mod 26 \\ &= c_1 + {c_2}^{-1} \mod 26 \\ D_{k,k'}(c) &:= |c - (k - k')| \\ &= |c + (k + (k')^{-1})^{-1}| \end{aligned}$$

Hence, the cryptosystem of the Caesar cypher satisfies the conditions 4, 5, 6. Therefore, it belongs to the class of VE. In fact, we calculate

$$\begin{aligned} D_{k,k'}(F(c_1, c_2)) &= |F(c_1, c_2) - (k - k')| \\ &= |c_1 - c_2 - k + k'| \\ &= |(p_1 + k) - (p_2 + k') - k + k'| \\ &= |p_1 - p_2| \\ &= V(p_1, p_2) \end{aligned}$$

Thus, it can be confirmed that a cryptosystem with a Caesar cypher belongs to the class of VE. □

3.2. SSO Based on VE

The Digital Identity Guidelines of the National Institute of Standards and Technology (NIST) [20] defines the authentication entities and prescribes the identity assurance standards for each process of identity proofing and authentication. These authentication entities are listed below.

User A person whose identity is to be verified using one or more authentication protocols.
Identity provider (IdP) A trusted entity that issues or registers a user's identity. An IdP may be an independent third party.
Verifier The entity that verifies the user's identity using an authentication protocol.
Relying party (RP) This entity relies upon the authentication information for user's identity sent from a verifier typically to process a transaction or permit access to information or a system.

To construct a simpler model than that defined by the Digital Identity Guidelines, we assume that IdP also plays the role of a verifier.

Furthermore, NIST defines the identity assurance levels (IALs) and authenticator assurance levels (AALs) as specifications for identity assurance. For additional information on IALs and AALs, please refer to NIST SP 800-63A and 800-63B, respectively [20].

IAL: *IAL refers to the robustness of the identity proofing process to confidently determine the identity of an individual. An IAL is selected to mitigate the potential identity proofing errors. (18)*
AAL: *AAL refers to the robustness of the authentication process as well as that of the binding between an authenticator and an individual's identifier. An AAL is selected to mitigate the potential authentication errors (i.e., a false claimant using a credential that does not rightfully belong to them). (18)*

The IALs and AALs are suggested as follows (Tables 2 and 3) [20].

Table 2. Identity Assurance Levels (IALs).

	Identity Assurance Level
IAL1	At IAL1, attributes, if any, are self-asserted or should be treated as self-asserted.
IAL2	At IAL2, either remote or in-person identity proofing is required. IAL2 requires the identification attributes to be verified in person or remotely using, at least, the procedures described in SP 800-63A.
IAL3	At IAL3, in-person identity proofing is required. Identification attributes must be verified by an authorized IdP representative by examining the physical documentation as described in SP 800-63A.

In our study, we selected IAL1 and AAL1 for identity assurance. In addition, we assumed that the authentication information provided to the RP and the personal information collected from the user are minimized according to the guidelines of [20]. Let Alice be a user, Bob be an IdP with a computation server S, and Charlie be an RP. The security policy of the proposed model can be summarized as follows.

- Alice accesses Charlie's service and does not reveal her identity to Charlie.
- Charlie provides service only to an authenticated user but does not handle the user's identity.

- Alice and Charlie trust Bob.
- The channel between Alice and Bob is a secure channel.
- The channel between Bob and Charlie is a secure channel.
- The channel between Alice and Charlie is an insecure channel.

Table 3. Authenticator Assurance Levels (AALs).

	AAL
AAL1	AAL1 provides some assurance that a claimant controls an authenticator registered to the subscriber. AAL1 requires single-factor authentication using a wide range of available authentication technologies. Successful authentication requires a claimant to prove the possession and control of the authenticator(s) through a secure authentication protocol.
AAL2	AAL2 provides high confidence that a claimant controls the authenticator(s) bound to the subscriber's account. Proof of possession and control of two distinct authentication factors are required through secure authentication protocol(s). Approved cryptographic techniques are required at AAL2 and above.
AAL3	AAL3 provides very high confidence that a claimant controls the authenticator(s) registered to the subscriber. Authentication at AAL3 is based on the proof of possession of a key through a cryptographic protocol. AAL3 is like AAL2, but also requires a "hard" cryptographic authenticator that provides verifier impersonation resistance.

The channels between the user (Alice), IdP (Bob), and RP (Charlie) are depicted in Figure 1. In Figure 1, the double-lined bars indicate a secure channel, while the single-lined bars indicate an insecure channel. Here, S is the computation server, which is independent of Alice, Bob, and Charlie as with [19]. S manages the database that cyphertexts is stored calculates cyphertexts with a map F. For the implementation of our algorithm, Bob contains both A and D, where A refers to a registration machine and D refers to a verification machine.

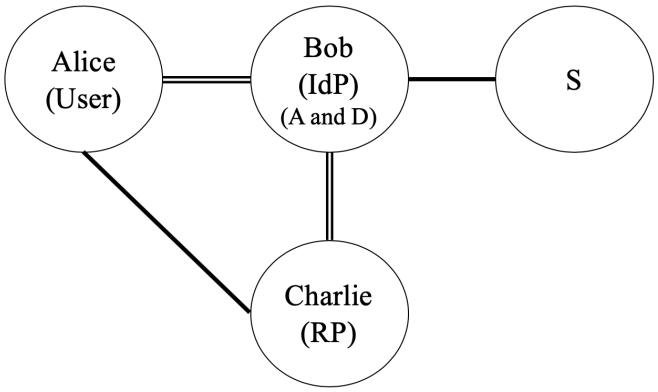

Figure 1. Channels between the user, IdP, and RP for our approach.

The one-time pad cryptosystem C_{otp}, which belongs the class of VE, can be defined as $C_{otp} = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where

$$\begin{aligned}
&\mathcal{P} = \mathcal{C} = \{0,1\}^n,\\
&\mathcal{K} = \{k \in \{0,1\}^n | \forall k, Pr(k) = \frac{1}{2^n}\},\\
&\mathcal{E} = \{E_k | E_k(p) = p \oplus k, \forall p \in \mathcal{P}, k \in \mathcal{K}\},\\
&\mathcal{D} = \{D_k | D_k(c) = c \oplus k, \forall c \in \mathcal{C}, k \in \mathcal{K}\}.
\end{aligned}$$

It should be noted that $Pr(k)$ is the appearance probability of k, and $\oplus$ indicates the bitwise exclusive OR operator.

Let $p_{A,i} \in \mathcal{P}$ be two plaintexts owned by Alice ($i = 1, 2$) , $k_{A,i}$ be keys, and $c_{A,i} \in \mathcal{C}$ be the two corresponding cyphertexts ($i = 1, 2$). Similarly, let $p_{C,i} \in \mathcal{P}$ be two plaintexts owned by Charlie, $k_{C,i}$ be keys, and $c_{C,i} \in \mathcal{C}$ be the two corresponding cyphertexts ($i = 1, 2$).

Registration step for Alice

Step 1 Alice sends $p_{A,1}$ to A
Step 2 A generates $k_{A,1}$ and computes $c_{A,1} = p_{A,1} \oplus k_{A,1}$
Step 3 A sends $c_{A,1}$ to S
Step 4 A send $k_{A,1}$ to D

Similar to Figure 1, the double-lined arrows indicate a secure channel, while the single-lined arrows indicate an insecure channel in Figures 2 and 3.

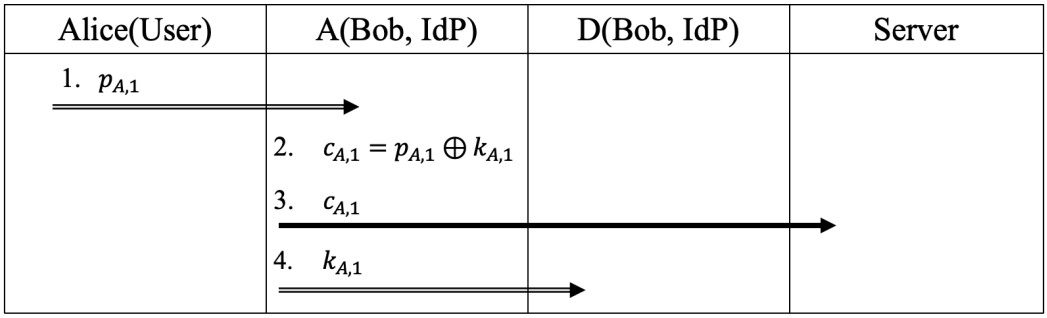

Figure 2. Registration step for Alice.

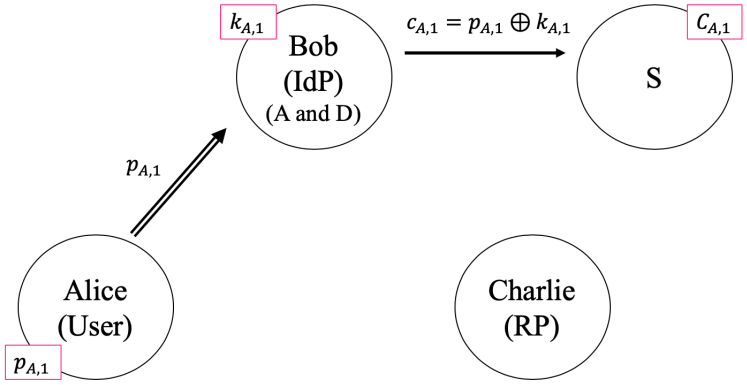

Figure 3. Registration step for Alice.

Registration step for Charlie

Step 1 Charlie sends $p_{C,1}$ to A
Step 2 A generates $k_{C,1}$ and computes $c_{C,1} = p_{C,1} \oplus k_{C,1}$
Step 3 A sends $c_{C,1}$ to S
Step 4 A sends $k_{C,1}$ to Charlie

Similar to Figure 1, the double-lined arrows and the single-lined arrows indicate a secure channel and insecure channel, respectively in Figures 4 and 5.

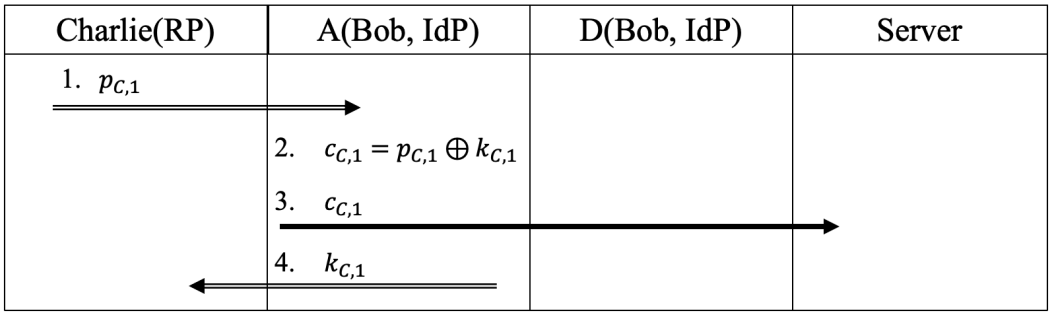

Figure 4. Registration step for Charlie.

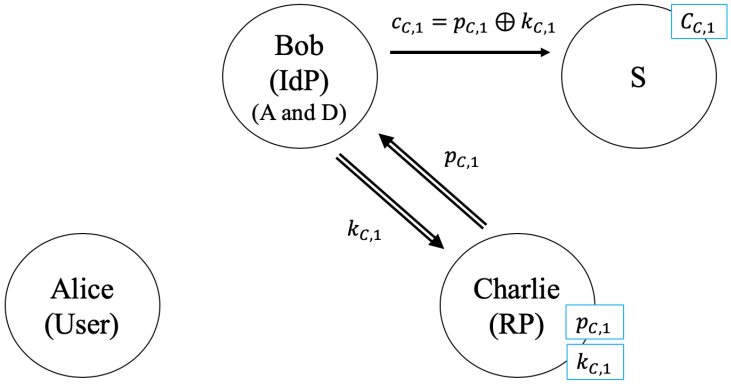

Figure 5. Registration step for Charlie.

Verification step

Step 1 Alice sends request for Charlie's service to Charlie
Step 2 Charlie generates $k_{C,2}$ and computes $c_{C,2} = p_{C,2} \oplus k_{C,2}$
Step 3 Charlie sends $c_{C,2}$ to Alice
Step 4 Alice computes $p_{c_{C,2}} = c_{C,2} \oplus p_{A,2}$ and sends $p_{c_{C,2}}$ to D
Step 5 D generates $k_{A,2}$ and computes $c_2 = p_{c_{C,2}} \oplus k_{A,2}$
Step 6 D sends c_2 to S
Step 7 S computes $c_3 = c_{A,1} \oplus c_{C,1} \oplus c_2$, where

$$\begin{aligned} c_3 &= c_{A,1} \oplus c_{C,1} \oplus c_2 \\ &= (p_{A,1} \oplus k_{A,1}) \oplus (p_{C,1} \oplus k_{C1}) \oplus (p_{c_{C,2}} \oplus k_{A,2}) \\ &= (p_{A,1} \oplus k_{A,1}) \oplus (p_{C,1} \oplus k_{C,1}) \oplus ((c_{C,2} \oplus p_{A,2}) \oplus k_{A,2}) \\ &= (p_{A,1} \oplus k_{A,1}) \oplus (p_{C,1} \oplus k_{C,1}) \oplus (((p_{C,2} \oplus k_{C,2}) \oplus p_{A,2}) \oplus k_{A,2}) \\ &= (p_{A,1} \oplus p_{A,2}) \oplus (p_{C,1} \oplus p_{C,2}) \oplus (k_{A,1} \oplus k_{A,2}) \oplus (k_{C,1} \oplus k_{C,2}) \end{aligned}$$

Step 8 S sends c_3 to D
Step 9 D computes $c_r = c_3 \oplus k_{A,1} \oplus k_{A,2}$, where

$$\begin{aligned} c_r &= c_3 \oplus k_{A,1} \oplus k_{A,2} \\ &= (p_{A,1} \oplus p_{A,2}) \oplus (p_{C,1} \oplus p_{C,2}) \oplus (k_{A,1} \oplus k_{A,2}) \oplus (k_{C,1} \oplus k_{C,2}) \oplus k_{A,1} \oplus k_{A,2} \\ &= (p_{A,1} \oplus p_{A,2}) \oplus (p_{C,1} \oplus p_{C,2}) \oplus (k_{C,1} \oplus k_{C,2}) \end{aligned}$$

Step 10 D sends c_r to Charlie

Step 11 Charlie computes $r = c_r \oplus k_{C,1} \oplus k_{C,2}$, where

$$\begin{aligned} r &= c_r \oplus k_{C,1} \oplus k_{C,2} \\ &= (p_{A,1} \oplus p_{A,2}) \oplus (p_{C,1} \oplus p_{C,2}) \oplus (k_{C,1} \oplus k_{C,2}) \oplus k_{C,1} \oplus k_{C,2} \\ &= (p_{A,1} \oplus p_{A,2}) \oplus (p_{C,1} \oplus p_{C,2}) \end{aligned}$$

Step 12 If $r = 0$, Charlie returns OK to Alice. Otherwise, Charlie returns NG.

Similar to Figure 1, the double-lined arrows and the single-lined arrows mean a secure channel and insecure channel in Figures 6 and 7. At step 12 in verification step, if $p_{C,1} \oplus p_{C,2} = 0$, the result of $p_{A,1} \oplus p_{A,2}$ can be known by only Charlie. Only Charlie is able to know the authentication result of Alice, however, he cannot obtain anything other than the authentication result. With this algorithm, even Bob who is an IdP trusted by Alice and Charlie, cannot know the authentication result.

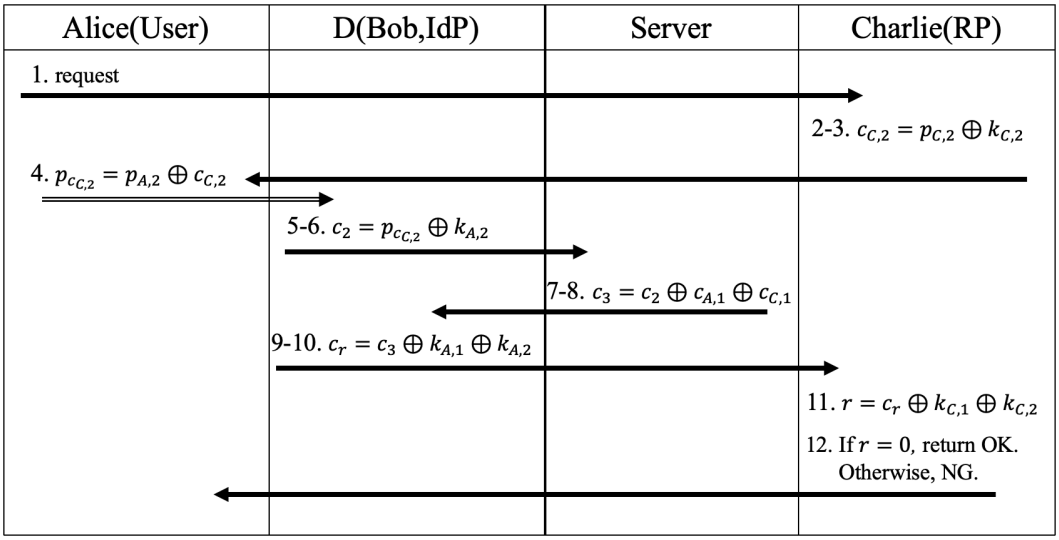

Figure 6. Verification step.

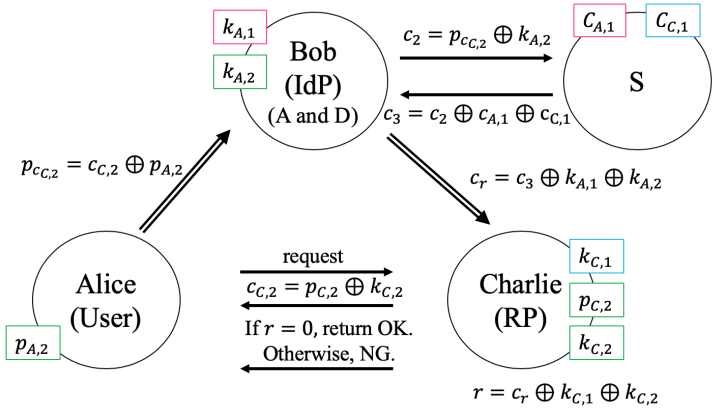

Figure 7. Verification step.

Let us consider the dependency of information from two sides; whether to be transmitted and who has the information.

At first, the following two Venn diagrams (Figures 8 and 9) show whether or not to transmit in the registration step and verification step, respectively. Here, the elements that belong to the intersection is the text to be transmitted.

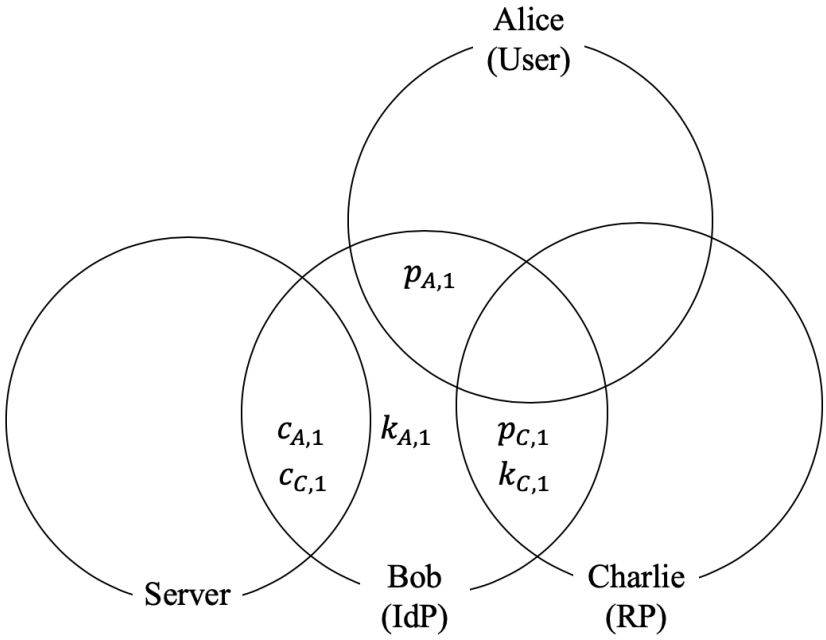

Figure 8. Venn diagram for registration step.

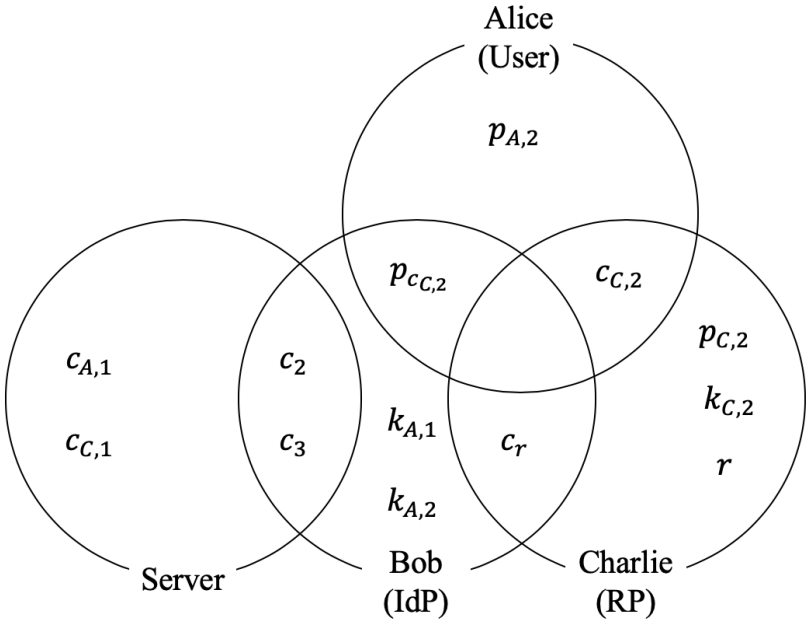

Figure 9. Venn diagram for verification step.

It can be seen from the Figure 8 that the plaintext of Alice $p_{A,1}$ is an element of the intersection of Alice and Bob, and the plaintext of Charlie $p_{C,1}$ and the key $k_{C,1}$ are elements of the intersection of Bob and Charlie. In contrast, it can be found from the Figure 9 that the plaintext of Alice $p_{A,2}$, the plaintext of Charlie $p_{C,2}$ and the key $k_{C,2}$ not in the intersection, namely, these texts are never sent as is to even Bob who is trusted by Alice and Charlie. For this result, it can be said that the channel between Alice and Bob and the channel between Bob and Charlie should be secure during the registration step. A Venn diagram summarizing both the registration and verification steps is shown in Figure 10.

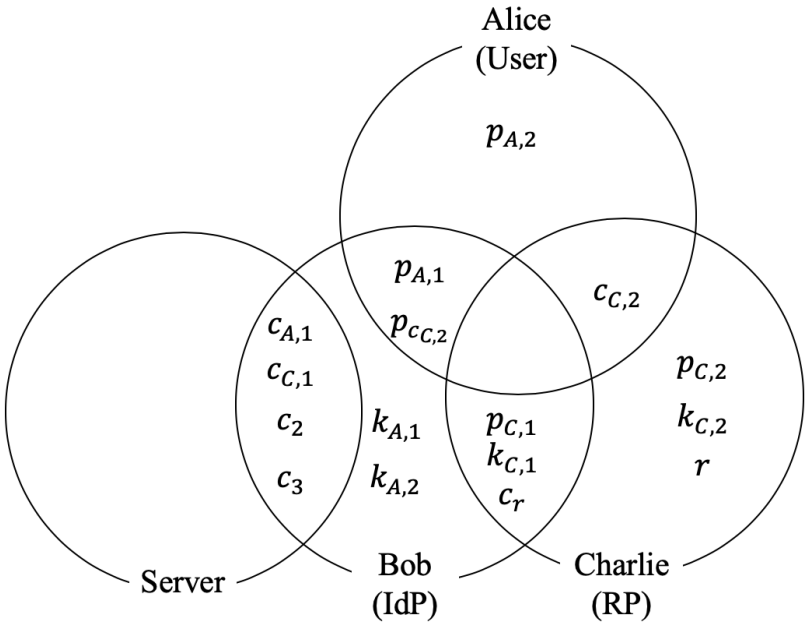

Figure 10. Venn diagram for all steps.

Next, Figure 11 shows the Venn diagram of the information held by each party.

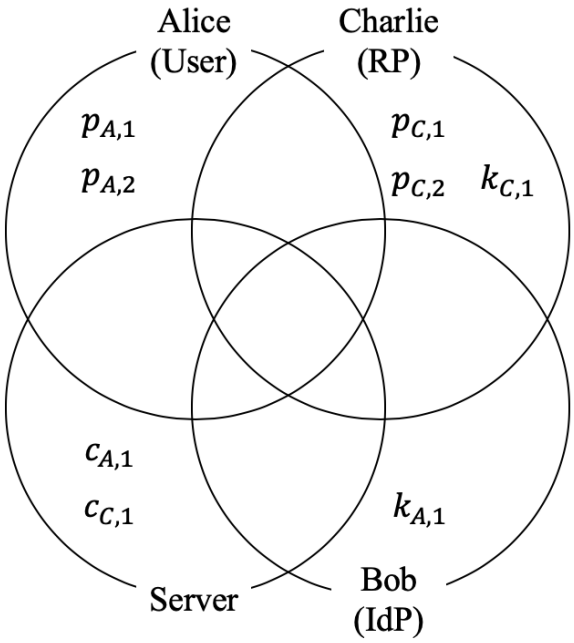

Figure 11. Venn diagram of thep.

It is obvious that all information is not in all intersections. Namely, there is no information held by two or more parties at the same time. Since all cyphertexts and keys are possessed by different parties, the server which possesses cyphertexts cannot decrypt.

Next, we describe the construction of the proposed SSO algorithm using the operator ∘ defined in Section 3.1.1.

Theorem 2. *If a cryptosystem such as that described in Theorem 1 is considered, the proposed SSO algorithm based on VE can be constructed.*

Proof. The cryptosystem $C = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, metric V, and maps F and D are given as follows.

- $\mathcal{P}, \mathcal{C}$, and $\mathcal{K}$ are the same arbitrary abelian groups, and $\mathcal{P}, \mathcal{C}, \mathcal{K} \subset G$, where G is a group closed by the operator $\circ$
- $\mathcal{E} = \{E_k | E_k(p) = p \circ k, \forall p \in \mathcal{P}, k \in \mathcal{K}\}$
- $\mathcal{D} = \{D_k | D_k(c) = c \circ k^{-1}, \forall p \in \mathcal{P}, k \in \mathcal{K}\}$
- $V(p_1, p_2) = p_1 \circ {p_2}^{-1}$
- $F(E_k(p_1), E_{k'}(p_2)) = E_k(p_1) \circ E_{k'}(p_2)^{-1}$
- $D_{k,k'}(c) = c \circ (k \circ (k')^{-1})^{-1}$

Let $p_{A,i} \in \mathcal{P}$ be two plaintexts owned by Alice, $k_{A,i}$ be the keys, and $E_{k_{A,i}}(p_{A,i}) = p_{A,i} \circ k_{A,i} = c_{A,i} \in \mathcal{C}$ be the two corresponding cyphertexts ($i = 1, 2$). Similarly, let $p_{C,i} \in \mathcal{P}$ be two plaintexts owned by Charlie, $k_{C,i}$ be the keys, and $E_{k_{C,i}}(p_{C,i}) = p_{C,i} \circ k_{C,i} = c_{B_i} \in \mathcal{C}$ be the two corresponding cyphertexts ($i = 1, 2$). It should be noted that the VE can be configured using only one cryptosystem C in this case.

Registration step for Alice

Step 1 Alice sends $p_{A,1}$ to A
Step 2 A generates $k_{A,1}$ and computes $c_{A,1} = E_{k_{A,1}}(p_{A,1}) = p_{A,1} \circ k_{A,1}$
Step 3 A sends $c_{A,1}$ to S
Step 4 A sends $k_{A,1}$ to D

Registration step for Charlie

Step 1 Charlie sends $p_{C,1}$ to A
Step 2 A generates $k_{C,1}$ and computes $c_{C,1} = E_{k_{C,1}}(p_{C,1}) = p_{C,1} \circ k_{C,1}$
Step 3 A sends $c_{C,1}$ to S
Step 4 A sends $k_{C,1}$ to Charlie

Verification step

Step 1 Alice sends request for Charlie's service to Charlie
Step 2 Charlie generates $k_{C,2}$ and computes $c_{C,2} = E_{k_{C,2}}(p_{C,2}) = p_{C,2} \circ k_{C,2}$
Step 3 Charlie sends $c_{C,2}$ to Alice
Step 4 Alice computes $p_{c_{C,2}} = p_{A,2} \circ c_{C,2}$ and sends $p_{c_{C,2}}$ to D
Step 5 D generates $k_{A,2}$ and computes $c_2 = E_{k_{A,2}}(p_{c_{C,2}})$, where

$$\begin{aligned} c_2 &= E_{k_{A,2}}(p_{c_{C,2}}) \\ &= p_{c_{C,2}} \circ k_{A,2} \\ &= (p_{A,2} \circ c_{C,2}) \circ k_{A,2} \\ &= (p_{A,2} \circ k_{A,2}) \circ c_{C,2} \\ &= E_{k_{A,2}}(p_{A,2}) \circ E_{k_{C,2}}(p_{C,2}) \\ &= c_{A,2} \circ c_{C,2} \end{aligned}$$

Step 6 D sends c_2 to S
Step 7 S computes $c_3 = c_{A,1} \circ c_{C,1} \circ {c_2}^{-1}$, where

$$\begin{aligned} c_3 &= c_{A,1} \circ c_{C,1} \circ c_2^{-1} \\ &= c_{A,1} \circ c_{C,1} \circ (c_{A,2} \circ c_{C,2})^{-1} \\ &= c_{A,1} \circ c_{C,1} \circ {c_{C,2}}^{-1} \circ {c_{A,2}}^{-1} \\ &= (c_{A,1} \circ {c_{A,2}}^{-1}) \circ (c_{C,1} \circ {c_{C,2}}^{-1}) \\ &= F(c_{A,1}, c_{A,2}) \circ F(c_{C,1}, c_{C,2}) \end{aligned}$$

Step 8 S sends c_3 to D
Step 9 D computes $c_r = D_{k_{A,1},k_{A,2}}(c_3)$, where

$$\begin{aligned} c_r &= D_{k_{A,1},k_{A,2}}(c_3) \\ &= D_{k_{A,1},k_{A,2}}(F(c_{A,1},c_{A,2}) \circ F(c_{C,1},c_{C,2})) \\ &= (F(c_{A,1},c_{A,2}) \circ F(c_{C,1},c_{C,2})) \circ (k_{A,1} \circ (k_{A,2})^{-1})^{-1} \\ &= (F(c_{A,1},c_{A,2}) \circ (k_{A,1} \circ (k_{A,2})^{-1})^{-1}) \circ F(c_{C,1},c_{C,2}) \\ &= D_{k_{A,1},k_{A,2}}(F(c_{A,1},c_{A,2})) \circ F(c_{C,1},c_{C,2}) \\ &= V(p_{A,1},p_{A,2}) \circ F(c_{C,1},c_{C,2}) \end{aligned}$$

Step 10 D sends c_r to Charlie
Step 11 Charlie computes $r = D_{k_{C,1},k_{C,2}}(c_r)$, where

$$\begin{aligned} r &= D_{k_{C,1},k_{C,2}}(c_r) \\ &= D_{k_{C,1},k_{C,2}}(V(p_{A,1},p_{A,2}) \circ F(c_{C,1},c_{C,2})) \\ &= (V(p_{A,1},p_{A,2}) \circ F(c_{C,1},c_{C,2})) \circ (k_{C,1} \circ (k_{C,2})^{-1})^{-1} \\ &= V(p_{A,1},p_{A,2}) \circ (F(c_{C,1},c_{C,2}) \circ (k_{C,1} \circ (k_{C2})^{-1})^{-1}) \\ &= V(p_{A,1},p_{A,2}) \circ D_{k_{C,1},k_{C,2}}(F(c_{C,1},c_{C,2})) \\ &= V(p_{A,1},p_{A,2}) \circ V(p_{C,1},p_{C,2}) \end{aligned}$$

Step 12 If $V(p_{C,1},p_{C,2}) = 0$, Charlie can obtain $V(p_{A,1},p_{A,2})$.
Step 13 Charlie confirms $V(p_{A,1},p_{A,2})$.

Hence, the SSO algorithm based on VE is constructed. □

Moreover, the following two corollaries are derived using Theorems 1 and 2.

Corollary 2. *A one-time pad can be constructed using the SSO algorithm based on VE.*

Proof. Considering [19], V, F, and D are composed of the operation $\oplus$. In the one-time pad cryptosystem, spaces for plaintexts, cyphertexts, and keys are as follows.

$$\begin{aligned} \mathcal{P} &= \mathcal{C} = \{0,1\}^n \\ \mathcal{K} &= \{k \in \{0,1\}^n | \forall k, P(k) = \frac{1}{2^n}\} \end{aligned}$$

${\mathbb{Z}_2}^n$ is an abelian group and isomorphic to $\{0,1\}^n$, namely,

$$\begin{aligned} \{0,1\}^n &\cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2 \\ &= \mathbb{Z}_2^n \end{aligned}$$

For all a, there exists a^{-1} that is a part of the one-time pad itself. Hence, $\{0,1\}^n$ is also an abelian group, and correspondingly, the one-time pad satisfies all conditions of Theorem 1. Therefore, the one-time pad can be successfully constructed using the SSO algorithm based on VE. □

Corollary 3. *The Caesar cypher cryptosystem can be constructed using the SSO algorithm based on VE.*

The proof is omitted because it is evident from Corollary 1 and Theorem 2.

4. Demonstration

We apply the QP-DYN algorithm [21] to our algorithm as a pseudorandom number generator. QP-DYN is a pseudorandom number generator that is faster than AES. Since the underlying VE-based authentication algorithm applies QP-DYN which has better speed performance than AES, we apply it in this study. For additional information on QP-DYN performance, please refer to [22]. Because any pseudorandom number generator can be used with the proposed algorithm, the time required for key generation is not discussed in this paper.

The specifications of the experimental environment are listed in Table 4.

Table 4. Experimental environment.

Aspect	Specification
Operating System	macOS High Sierra 10.13.5
CPU	Intel Core i7 1.7 GHz
Memory	8 GB
Language	Python 2.7

The registration information for Alice and Charlie is presented in Tables 5 and 6, respectively.

The verification is presented in Table 7. In this case, the plaintexts of Alice and Charlie at the verification are identical with the plaintexts used at the registration, respectively. Namely, $p_{A,2} = p_{A,1}$ and $p_{C,2} = p_{C,1}$.

The verification times are listed in Table 8 and graphically shown in Figure 12. In the proposed algorithm, the speed from the time Charlie received the request sent by Alice in the verification step 1 to the output of the final calculation r in the verification step was measured. Table 8 and Figure 12 show the averages of 100 measurements taken for each text length. It is evident from the results above that increasing the length of the plaintext leads to an increase in the execution time; however, it is noteworthy that a plaintext of 8192 bits can be processed in less than 1 ms.

Table 5. Registration information for Alice.

Type	Variable	Value
Plaintext	$p_{A,1}$	1100110110101000101110111101000100010011101111111001101110110100010101000010111001001100011111011000000011111011100000000100001
Key	$k_{A,1}$	1100101111111010010010011011101010110111011001001111001111010000111001001111000011011000011010111011111100011100101001100100100
Ciphertext	$c_{A,1}$	0011010110011001110000000011111110100101010110110111110000010101100111011100111111111110111010011010101101100001011001110110011

Table 6. Registration information for Charlie.

Type	Variable	Value
Plaintext	$p_{C,1}$	11100111111100001010100100001110 01111000011111110101110010000110 10111011011010000110100100000000 11111000010101000101101010110000
Key	$k_{C,1}$	10001000011100101001110011110010 01011100101000101011111100100111 01100110101000100111111010110101 01011101101111111011001101000000
Ciphertext	$c_{C,1}$	01101111100000100011010111111100 00100100110111011110001110100001 11011101110010100001011110110101 10100101111010111110100111110000

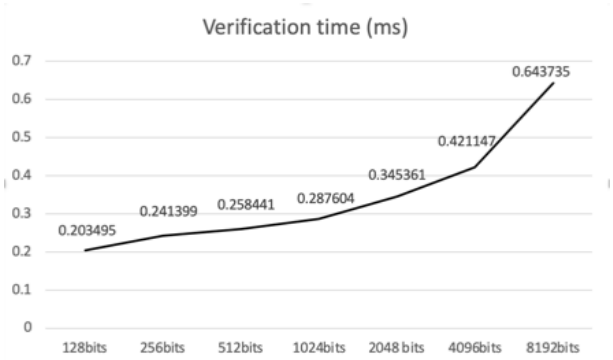

Figure 12. Speed test results for verification step.

Table 7. Verification information.

Type	Variable	Value
One-time key for Charlie	$k_{C,2}$	11100101010100111100101001000100 10111111110010101110010110101100 10100100000011000000011111000111 00011111001101110110111001001100
Ciphertext	$c_{C,2}$	00000010101000110110001101001010 11000111101101011011100100101010 00011111011001000110111011000111 11100111011000110011010011111100
Plaintext of Alice with cyphertext $c_{C,2}$	$p_{c_{C,2}}$	11111100110000001110101011001111 11010100000010100010001010011110 01001011010010100010001010111010 01100111000111101111010010111110
One-time key for Alice	$k_{A,2}$	10100010100000000111100110001000 10001010101010000100000111010011 00111110001100110000001110011100 01101001011100010110101101000111
Ciphertext	c_2	01011110010000001001001101000111 01011110101000100110001101001101 01110101011110010010000100100110 00001110011011111001111111111001
Encrypted result	c_3	00000100010110110110011010000100 11011111001001001111110011111001 00110101011111001100101101000000 11110100111001010001000101101111
Result for Alice with encrypted result for Charlie	c_r	01101101001000010101011010110110 11100011011010000101101010001011 11000010101011100111100101110010 01000010100010001101110100001100
Result	r	0

Table 8. Speed test results with different plaintext lengths.

Length of Plaintext	Verification Time
128 bits	0.203495 ms
256 bits	0.241399 ms
512 bits	0.258441 ms
1024 bits	0.287604 ms
2048 bits	0.345361 ms
4096 bits	0.421147 ms
8192 bits	0.643735 ms

5. Discussion

In this section, we discuss the robustness of the algorithm against attacks as well as present a special case of the algorithm and compare the proposed algorithm with Kerberos, OpenID, and SAML. Moreover, we discuss the impact on the customers and businesses.

5.1. Robustness against Attacks

5.1.1. Theoretical and Classical Attack

Here, we discuss the robustness of the algorithm against theoretical and classical attacks.

First, the results of a cyphertext-only attack (COA) against our algorithm and the case wherein the authentication result r becomes 0 are discussed. COA is the attack model that seeks plaintext using only cyphertext, and also called the known cyphertext attack (KCA). Each cyphertext is listed in Table 9, where $p_{A,1}$ and $p_{A,2}$ are the plaintexts of Alice at the registration and verification steps, respectively. Similarly, $p_{C,1}$ and $p_{C,2}$ are the plaintexts of Charlie at the registration and verification steps, respectively.

Table 9. Appearing cyphertexts.

Ciphertext	Contents
$c_{A,1}$	$p_{A,1} \oplus k_{A,1}$
$c_{C,1}$	$p_{C,1} \oplus k_{C,1}$
$c_{C,2}$	$p_{C,2} \oplus k_{C,2}$
$p_{c_{C,2}}$	$p_{A,2} \oplus c_{C,2}$ $= p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
c_2	$p_{c_{C,2}} \oplus k_{A,2}$ $= p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
c_3	$c_2 \oplus c_{A,1} \oplus c_{C,1}$ $= p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2} \oplus p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
c_r	$c_3 \oplus k_{A,1} \oplus k_{A,2}$ $= p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus p_{A,1} \oplus p_{C,1} \oplus k_{C,1}$

It should be noted that in our study, $p_{c_{C,2}}$ and c_r are targeted via the cyphertext-only attack; however, they are sent to A and Charlie via a secure channel, respectively.

We show the attack by changing the number of elements to be extracted from the set of cyphertexts $\{c_{A,1}, c_{C,1}, c_{C,2}, p_{c_{C,2}}, c_2, c_3, c_r\}$.

Table 10 shows the case when the number of elements to be extracted is two. In this case, $p_{A,2}$ and $k_{A,2}$ can be stolen during the combinations $(c_{C,2}, p_{c_{C,2}})$ and $(p_{c_{C,2}}, c_2)$, respectively. However, there is no need to be concerned regarding the security of the system in this case because $p_{c_{C,2}}$ is sent via a secure channel.

Table 10. Attack scenario wherein two elements are extracted.

Combination	Calculation
$c_{A,1} \oplus c_{C,1}$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_{A,1} \oplus c_{C,2}$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus p_{c_{C,2}}$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_3$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$
$c_{C,1} \oplus c_{C,2}$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,1} \oplus p_{c_{C,2}}$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,1} \oplus c_2$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_r$	$p_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,2} \oplus p_{c_{C,2}}$	$p_{A,2}$
$c_{C,2} \oplus c_2$	$p_{A,2} \oplus k_{A,2}$
$c_{C,2} \oplus c_3$	$p_{A,2} \oplus k_{A,2} \oplus p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_{C,2} \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$p_{c_{C,2}} \oplus c_2$	$k_{A,2}$
$p_{c_{C,2}} \oplus c_3$	$k_{A,2} \oplus p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$p_{c_{C,2}} \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_2 \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_2 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$c_3 \oplus c_r$	$k_{A,1} \oplus k_{A,2}$

Table 11 shows the case when the number of elements to be extracted is three.

Table 11. Attack scenario wherein three elements are extracted.

Combination	Calculation
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2}$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}}$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus p_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_3$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_r$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}}$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus c_3$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus k_{A,1}$
$c_{A,1} \oplus p_{c_{C,2}} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus p_{c_{C,2}} \oplus c_3$	$p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$c_{A,1} \oplus p_{c_{C,2}} \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus k_{A,1}$
$c_{A,1} \oplus c_2 \oplus c_3$	$p_{C,1} \oplus k_{C,1}$
$c_{A,1} \oplus c_2 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}}$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus c_2$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus c_r$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus k_{A,1}$
$c_{C,1} \oplus p_{c_{C,2}} \oplus c_2$	$p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$c_{C,1} \oplus p_{c_{C,2}} \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus k_{A,2}$
$c_{C,1} \oplus p_{c_{C,2}} \oplus c_r$	$p_{A,1}$
$c_{C,1} \oplus c_2 \oplus c_3$	$p_{A,1} \oplus k_{A,1}$
$c_{C,1} \oplus c_2 \oplus c_r$	$p_{A,1} \oplus k_{A,2}$
$c_{C,1} \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus k_{A,1} \oplus k_{A,2}$
$c_{C,2} \oplus p_{c_{C,2}} \oplus c_2$	$p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{C,2} \oplus p_{c_{C,2}} \oplus c_3$	$p_{C,2} \oplus k_{C,2} \oplus k_{A,2} \oplus p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_{C,2} \oplus p_{c_{C,2}} \oplus c_r$	$p_{C,2} \oplus k_{C,2} \oplus p_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_{C,2} \oplus c_2 \oplus c_3$	$p_{C,2} \oplus k_{C,2} \oplus p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_{C,2} \oplus c_2 \oplus c_r$	$p_{C,2} \oplus k_{C,2} \oplus p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$c_{C,2} \oplus c_3 \oplus c_r$	$p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_2 \oplus c_3 \oplus c_r$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$

In this case, $p_{A,1}$ can be stolen during the combination $(c_{C,1}, p_{c_{C,2}}, c_r)$. However, similar to the case of the extraction of two elements, $p_{c_{C,2}}$ is not compromised.

Table 12 shows the case when the number of elements to be extracted is four.

Table 12. Attack scenario wherein four elements are extracted.

Combination	Calculation
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}}$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus c_3$	$p_{A,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus c_r$	$p_{A,2} \oplus k_{A,1}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}} \oplus c_3$	$k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}} \oplus c_r$	$k_{A,1}$
$c_{A,1} \oplus c_{C,1} \oplus c_2 \oplus c_3$	0
$c_{A,1} \oplus c_{C,1} \oplus c_2 \oplus c_r$	$k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_3$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$
$c_{A,1} \oplus c_{C,2} \oplus c_2 \oplus c_3$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus c_{C,2} \oplus c_2 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_r$	$p_{A,1} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,1} \oplus c_{C,2} \oplus c_2 \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,1} \oplus c_{C,2} \oplus c_2 \oplus c_r$	$p_{A,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{C,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{A,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{C,1} \oplus p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$
$c_{C,1} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{A,1} \oplus k_{A,1} \oplus p_{A,2} \oplus p_{C,1} \oplus k_{C,1}$
$c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{A,1} \oplus p_{A,2} \oplus p_{C,1} \oplus k_{C,1} \oplus k_{A,2}$
$c_{C,2} \oplus p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{A,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{C,2} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,2} \oplus k_{A,1}$
$p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$k_{A,1}$

In this case, there is a possibility that $k_{A,2}$ and $k_{A,1}$ can be stolen. However, it should be noted that both include $p_{c_{C,2}}$ in the calculation.

Table 13 shows the case when the number of elements to be extracted is five.

In this case, there is a possibility that $p_{A,1}$ and $p_{A,1} \oplus p_{A,2}$ can be stolen. However, $p_{c_{C,2}}$ and c_r are included in the calculation.

Table 14 shows the case when the number of elements to be extracted is six.

In this case, $p_{A,2}$ can be stolen. However, $p_{A,2}$ is protected because $p_{c_{C,2}}$ is included in the calculation.

Even when the calculation is performed using all cyphertexts, neither Alice nor Bob's plaintext information is compromised (See Table 15).

Table 13. Attack scenario wherein five elements are extracted.

Combination	Calculation
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2$	$p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_3$	$p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_r$	$p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus c_2 \oplus c_3$	$p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus c_2 \oplus c_r$	$p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus p_{C,2} \oplus k_{C,2}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{A,2} \oplus p_{C,1} \oplus k_{C,1} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,2} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{A,2}$
$c_{A,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{A,1} \oplus p_{A,2} \oplus k_{A,1}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{A,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{C,1} \oplus c_{C,2} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus k_{A,1}$
$c_{C,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus k_{A,1}$
$c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$

Table 14. Attack scenario wherein six elements are extracted.

Combination	Calculation
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3$	$p_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_r$	$p_{A,2} \oplus k_{A,1} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{A,2} \oplus k_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{A,2}$
$c_{A,1} \oplus c_{C,1} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1}$
$c_{A,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,2} \oplus k_{C,2}$
$c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2} \oplus k_{A,1}$

Table 15. Attack scenario wherein all elements are extracted.

Combination	Calculation
$c_{A,1} \oplus c_{C,1} \oplus c_{C,2} \oplus p_{c_{C,2}} \oplus c_2 \oplus c_3 \oplus c_r$	$p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_{C,2} \oplus k_{C,2}$

In addition to the cyphertext only attack, we present our consideration of the classical and theoretical attacks (known plaintext attack (KPA), chosen plaintext attack (CPA), chosen cyphertext attack (CCA1), and adaptive chosen cyphertext attack (CCA2)) against our algorithm.

KPA is the attack model that the plaintext is obtained from the cyphertext under the condition that the cyphertext corresponding to the known plaintext can be obtained. In KPA, all cyphertexts are assumed using the same private key for encryption. This attack does not hold against our algorithm, where all cyphertexts are encrypted with different keys.

CPA is the attack model that obtains a plaintext from a cyphertext under the condition that a cyphertext corresponding to an arbitrary plaintext can be obtained. We have already discussed about CPA against the underlying VE-based authentication algorithm of our SSO algorithm in previous paper [19]. An attacker could get plaintext if a stream cypher was used instead of a one-time pad. However, since the one-time pad is applied to our algorithm, the algorithm is secure by Shanon's perfect secrecy.

CCA1 and CCA2 are the attack model that obtains a plaintext from a certain cyphertext under the condition that a plaintext corresponding to an arbitrary cyphertext excluding the cyphertext to

be decrypted can be obtained. From this obtained information, the attacker attempts to recover the private key used for decryption. However, similar to CPA, the used key for encryption cannot be obtained by the attacker because we apply a one-time pad to the algorithm.

Let us assume that the condition $r = 0$ is realized at Step 11 of the verification process in Theorem 1. The calculation step $r = V(p_{A,1}, p_{A,2}) \circ V(p_{C,1}, p_{C,2})$ can be considered as

$$\begin{aligned} r &= V(p_{A,1}, p_{A,2}) \circ V(p_{C,1}, p_{C,2}) \\ &= V(V(p_{A,1}, p_{A,2}), V(p_{C,1}, p_{C,2})) \end{aligned}$$

Here, $V(a, b) = a \circ b^{-1}$. From the above equation, the condition that r is equal to 0 is represented by $V(p_{A,1}, p_{A,2}) = V(p_{C,1}, p_{C,2})$. If $V(p_{A,1}, p_{A,2}) = V(p_{C,1}, p_{C,2}) = 0$, then authentication is successful. However, there is a possibility that $V(p_{A,1}, p_{A,2}) = V(p_{C,1}, p_{C,2}) \neq 0$ and $r = 0$; nevertheless, the occurrence probability can be reduced by increasing the size of the plaintext.

5.1.2. Man-in-the-Middle Attack

Man-in-the-middle attack (MITM) is an active attack in which the attacker secretly relays the information communicated by two parties who believe that they are communicating directly with each other, and then eavesdrops or alters it. We discuss the robustness of our proposed algorithm against the MITM, which is a well-known attack for Kerberos, OpenID, and SAML.

MITM effectively attacks against the algorithms using the public key cryptography and agreement; however, the public key cryptography and agreement are not required in our algorithm. Therefore, MITM cannot hold against our algorithm. The three major algorithms can avoid man-in-the-middle attacks by a signature to the token and using a secure channel such as SSL/TLS.

Let us consider that attacker Eve masqueraded as Charlie who is RP in the verification step of our algorithm (See Figures 13 and 14). We assume that Eve wants to obtain Alice's authentication information but does not possess the registered plaintext of Charlie $p_{C,1}$ and key $k_{C,1}$. Although the channel between Bob and Charlie is actually secure, we assume that Eve communicates with Alice on the same channel as Charlie. Suppose that the condition setting is the same as the algorithm proposed in Section 2.

Verification step

Step 1 Alice sends a request to Eve, impersonating Charlie's service.
Step 2 Eve generates k_E and computes $c_E = E_{k_E}(p_E) = p_E \oplus k_E$
Step 3 Eve sends c_E to Alice
Step 4 Alice computes $p_{c_E} = p_{A,2} \oplus c_E$ and sends p_{c_E} to D
Step 5 D generates $k_{A,2}$ and computes $c_2 = p_{c_E} \oplus k_{A,2}$
Step 6 D sends c_2 to S
Step 7 S computes $c_3 = c_{A,1} \oplus c_{C,1} \oplus c_2$, where

$$\begin{aligned} c_3 &= c_{A,1} \oplus c_{C,1} \oplus c_2 \\ &= (p_{A,1} \oplus k_{A,1}) \oplus (p_{C,1} \oplus k_{C1}) \oplus (p_{c_E} \oplus k_{A,2}) \\ &= (p_{A,1} \oplus k_{A,1}) \oplus (p_{C,1} \oplus k_{C,1}) \oplus ((c_E \oplus p_{A,2}) \oplus k_{A,2}) \\ &= p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_E \oplus k_E \oplus p_{A,2} \oplus k_{A,2} \end{aligned}$$

Step 8 S sends c_3 to D
Step 9 D computes $c_r = c_3 \oplus k_{A,1} \oplus k_{A,2}$, where

$$\begin{aligned} c_r &= c_3 \oplus k_{A,1} \oplus k_{A,2} \\ &= p_{A,1} \oplus k_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_E \oplus k_E \oplus p_{A,2} \oplus k_{A,2} \oplus k_{A,1} \oplus k_{A,2} \\ &= p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_E \oplus k_E \oplus p_{A,2} \end{aligned}$$

Step 10 D sends c_r to Eve who impersonates Charlie
Step 11 Eve computes $r = c_r \oplus c_E$, where

$$\begin{aligned} r &= c_r \oplus c_E \\ &= p_{A,1} \oplus p_{C,1} \oplus k_{C,1} \oplus p_E \oplus k_E \oplus p_{A,2} \oplus p_E \oplus k_E \\ &= (p_{A,1} \oplus p_{A,2}) \oplus (p_{C,1} \oplus k_{C,1}) \\ &= (p_{A,1} \oplus p_{A,2}) \oplus c_{C,1} \end{aligned}$$

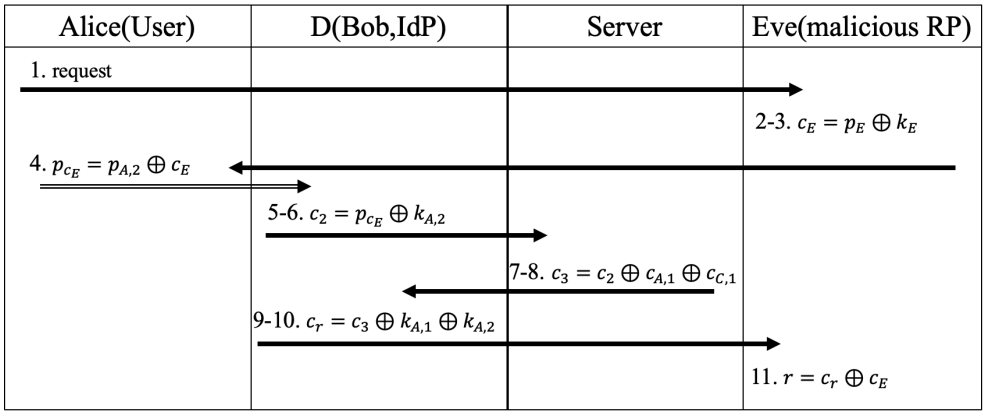

Figure 13. Man-in-the-Middle Attack (MITM) in verification step.

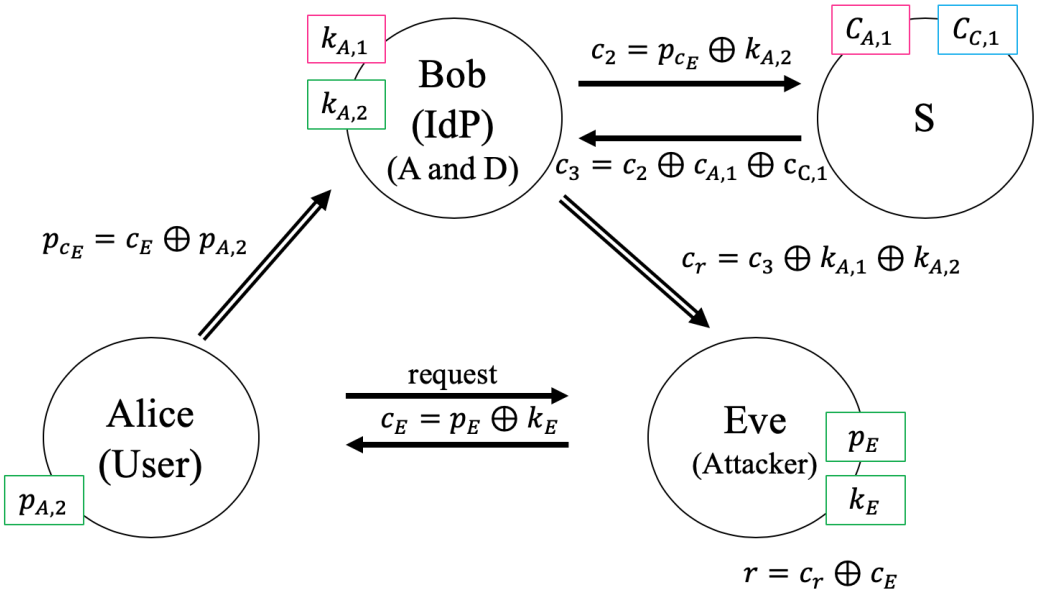

Figure 14. MITM in verification step.

If Eve could obtain $c_{C,1}$ in the registration step, she can compute the authentication information of Alice with r and $c_{C,1}$ as follows

$$\begin{aligned} r \oplus c_{c,1} &= (p_{A,1} \oplus p_{A,2}) \oplus c_{C,1} \oplus c_{C,1} \\ &= p_{A,1} \oplus p_{A,2} \end{aligned}$$

Eve can obtain only the distance between two plaintexts of Alice $p_{A,1}$ and $p_{A,2}$. In fact, because the channel between Bob and Charlie is a secure channel, this attack does not hold.

Moreover, consider the case in which Bob who is an IdP trusted by Alice and Charlie is impersonated by Eve who is a malicious third party. In this case, Eve can obtain plaintext from both Alice and Charlie at the registration step. Bob's reliability as a security policy is very important because the protocol does not hold. That is, the channel between Alice and Bob and the channel between Bob and Charlie must be secure or directly accessible.

5.1.3. Security Analysis

Here, we show security analyses of our algorithm using Proverif [23] in Tables 16–18.

Table 16. Security analysis of registration step for Alice.

```
Process 0 (that is, the initial process):
{1}new ka1: key;
(
{2}!
{3}out(atob, pa1)
) | (
{4}!
{5}in(atob, x: bitstring);
{6}out(btos, senc(x,ka1))
)

– Query not attacker(pa1[]) in process 0.
Completing...
Starting query not attacker(pa1[])
RESULT not attacker(pa1[]) is true.

_____
Verification summary:

Query not attacker(pa1[]) is true.

_____
```

Table 17. Security analysis of registration step for Charlie.

```
Process 0 (that is, the initial process):
{1}new kc1: key;
(
{2}!
{3}out(ctob, pc1)
) | (
{4}!
{5}in(ctob, x: bitstring);
{6}out(btos, senc(x,kc1));
{7}out(btoc, kc1)
)

– Query not attacker(pc1[]) in process 0.
Completing...
Starting query not attacker(pc1[])
RESULT not attacker(pc1[]) is true.

_____
Verification summary:

Query not attacker(pc1[]) is true.

_____
```

Table 18. Security analysis of verification step.

```
Process 0 (that is, the initial process):
{1}new ka2: key;
{2}new kc2: key;
(
{3}!
{4}out(atoc, request);
{5}in(ctoa, cc2: bitstring);
{6}out(atob, g(cc2,pa2))
) | (
{7}!
{8}in(atoc, req: bitstring);
{9}out(ctoa, senc(pc2,kc2));
{10}in(btoc, cr: bitstring);
{11}out(ctoa, result)
) | (
{12}!
{13}in(atob, cp: bitstring);
{14}out(btos, senc(cp,ka2));
{15}in(btos, c3: bitstring);
{16}out(btoc, d(c3,ka1,ka2))
) | (
{17}in(btos, c2: bitstring);
{18}out(stob, f(c2,ca1,cc1))
)

– Query not attacker(pa2[]) in process 0.
Completing...
Starting query not attacker(pa2[])
RESULT not attacker(pa2[]) is true.
– Query not attacker(pc2[]) in process 0.
Completing...
Starting query not attacker(pc2[])
RESULT not attacker(pc2[]) is true.

--------------------------------------------------------------------------------
Verification summary:

Query not attacker(pa2[]) is true.

Query not attacker(pc2[]) is true.

--------------------------------------------------------------------------------
```

From the above three tables, it is obvious to say that our algorithm maintains security in all phases.

5.2. Comparison

Table 19 presents the comparisons between the proposed VE-based SSO algorithm and the Kerberos-based, OpenID, and SAML implementations.

Table 19. Comparison of the proposed VE-based SSO implementation with other major SSO implementations.

	Kerberos-Based [8,9]	**OpenID [11]**	**SAML [10,24]**	**Proposed VE-Based SSO**
Independence of IdP and RP	Not independent	Independent	Not independent	Independent
Cryptosystem	AES	Diffie-Hellman	RSA	One-time pad
Usable identities	ID/Password	ID/Password	ID/Password	Any

The first line compares the independence of IdP and RP. OpenID and our algorithm are assumed to be independent between IdP and RP; however, Kerberos and SAML are not independent. Because the independency depends on the requirements of the application used, it cannot be said which algorithm is better.

The second line compares the cryptographic systems used. The cryptosystems used in major SSO implementations are considered to be computationally secure cryptosystems. In contrast, VE-based SSO algorithms are based on the one-time pad, which is a theoretically secure information cryptosystem. Furthermore, our proposed algorithm and Kerberos are based on secret key cryptography, while the other two algorithms are based on public key cryptography (including agreement). Obviously, the speed is higher when secret key cryptography is used when comparing only cryptosystems. However, in secret key cryptography, it is necessary to share the key in advance, so it cannot be clearly said which algorithm is faster. In the registration step of our algorithm, although there is a key distribution from Bob who is IdP, to Charlie who is RP, the key is distributed as it is via a secure channel. The encryption using a cryptographic system such as public key agreement is not required. Because Bob never holds Charlie's key, the key management burden of Bob is small. In addition, AES which is applied to Kerberos is applicable to our implementation.

As listed in the third line in the table above, any identity can be applied to our algorithm; in contrast, only the ID and password can be applied using other implementations. In our algorithm, because usable personal information is arbitrary, password attacks such as pass the hash attack, password list attack, and password spray attack cannot be performed against the algorithm unless the pair of ID and password is applied.

5.3. The Impact on Businesses

As mentioned in the introduction, the use of SSO leads to five benefits: Reduction in help desk costs, improved in customer satisfaction, boost in productivity, improvement in compliance and security capabilities, and B2B collaboration. In addition to these benefits, we discuss the impact of our algorithm on our business and our customers.

First, our proposed algorithm can be expected to exhibit a different personal information management method than existing algorithms. In the VE-based authentication algorithm, which is the basis of the SSO algorithm proposed here, the server that manages the database cannot obtain any authentication information as well as personal information and keys. Moreover, our algorithm does not need to store personal information in local storage or the database of IdP and RP and enables information decentralization. The VE-based SSO algorithm proposed in this study differs from other SSO algorithms in that the information lists held by the IdP or RP and the encrypted personal information can be managed separately.

Next, the VE based-SSO algorithm enables the application of any personal information in contrast with the algorithm of a typical SSO which is based on a pair of ID and password, as compared in the above Section 5.2. Because the amount of personal information that users can select other than ID, such as biometric information, increases, it may be possible for the users to protect their own personal information by selecting the information that they want to use. Moreover, our assumption is the construction of a system with independent RP and IdP, which states that only the authentication result is disclosed to the RP and that the authentication result is not disclosed to the IdP. In other words, IdP cannot obtain any information in the verification step, and RP cannot obtain any information other than the authentication information as to whether the plaintexts of the user match. Because users can avoid providing information that they do not wish to disclose, it may reduce the possibility of the unintended collection of user information by GAFA, which is a problem in [14].

Finally, in addition to the benefits of using SSO, our algorithm has the potential to realize SSO that solves the current problems such as unintended collection of user information and the hesitancy of disclosure of their personal information.

6. Conclusions

In this study, we constructed a VE-based algorithm that enables users to securely provide authentication information to a service provider and achieve SSO. By using the proposed algorithm, the personal information of the user is not required to be unnecessarily disclosed and used without their permission. Moreover, the algorithm can be used with a pair of ID and password as well as other identity information, including biometric information, and unique numbers known by the user, among others. Based on our experimental results, the proposed algorithm takes less than 1 ms even for a plaintext with a length of 8192 bits. Our algorithm maintains security against classical and theoretical attacks. The security of our algorithm against classical and theoretical attacks is maintained. Also, the man-in-the-middle attack, which is one of the most popular methods against major SSO implementations is impossible against our algorithm because our algorithm does not require public key agreement.

7. Patents

The VE-based authentication algorithm proposed in [19] is patented [25]. Patents for the SSO algorithm proposed in this study are currently pending [26].

Author Contributions: Conceptualization, S.I., M.K.; validation, M.K.; writing—original draft preparation, M.K., S.I.; writing—review and editing, M.K., S.I., Editage; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Internet World Stats. Available online: https://www.internetworldstats.com/stats.htm (accessed on 31 March 2020).
2. Hu, J.; Sun, Q.; Chen, H. Application of single sign-on (SSO) in digital campus. In Proceedings of the Third IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT 2010), Beijing, China, 26–28 October 2010; IEEE: Piscataway, NJ, USA, 2011; pp. 725–727.
3. Single Sign-on Market by Type (Enterprise, Federated & Web-Based, Windows Integrated), Organization Size (Small & Medium Enterprises, Large Enterprises), Deployment Mode (Cloud, On-Premises), Vertical, Region—Global Forecast to 2021. Markets and Markets. Available online: https://www.marketsandmarkets.com/Market-Reports/single-sign-on-market-83280444.html (accessed on 28 May 2020).
4. Villanueva, J. 5 Big Business Benefits of Using SSO (Single Sign-On). Managed File Transfer and Network Solutions. Available online: https://www.jscape.com/blog/bid/104856/5-Big-Business-Benefits-of-Using-SSO-Single-Sign-On (accessed on 14 May 2020).
5. Radha, V.; Reddy, D.H. A survey on single sign-on techniques. *Proc. Technol.* **2012**, *4*, 134–139. [CrossRef]
6. Bazaz, T.; Khalique, A. A review on single sign on enabling technologies and protocols. *Int. J. Comput. Appl.* **2016**, *151*, 18–25. [CrossRef]
7. De Clercq, J. Single sign-on architectures. In Proceedings of the International Conference on Infrastructure Security, Heidelberg, Germany, 1 October 2002; Springer: Berlin, Germany, 2002; pp. 40–58.
8. Miller, S.P.; Neuman, B.C.; Schiller, J.I.; Saltzer, J.H. Project Athena Technical Plan. In *Kerberos Authentication and Authorization System*, Massachusetts Institute of Technology: Cambridge, MA, USA,1988.
9. Neuman, B.C.; Ts'o, T. Kerberos: An authentication service for computer networks. *IEEE Commun. Mag.* **1994**, *32*, 33–38. [CrossRef]
10. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 OASIS Standard, OASIS, 2005. Available online: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf (accessed on 6 April 2020).
11. OpenID Foundation Website. Available online: http://openid.net/ (accessed on 6 April 2020).
12. OAuth Website. Available online: https://oauth.net/2/ (accessed on 28 May 2020).

13. Sun, S.T.; Pospisil, E.; Muslukhov, I.; Dindar, N.; Hawkey, K.; Beznosov, K. Beznosov KWhat makes users refuse web single sign-on? An empirical investigation of OpenID. In Proceedings of the Seventh Symposium on Usable Privacy and Security, Pittsburgh, PA, USA, 20–22 July 2011.
14. Horie, S. Keeping Anonymity at the Consumer Behavior on the Internet: Proof of Sacrifice. *Comput. Ethics-Philos. Enq. (CEPE) Proc.* **2019**, *2019*, 5.
15. Yang, F.; Manoharan, S. A security analysis of the OAuth protocol. In Proceeding of the 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Canada, 27–29 August 2013
16. Tsyrklevich, E.; Tsyrklevich, V. Single sign-on for the internet: A security story. 2007. Available Online: https://bit.ly/2UCJZDo (accessed on 11 June 2020)
17. Groß, T. Security analysis of the SAML single sign-on browser/artifact profile. In Proceeding of the 19th Annual Computer Security Applications Conference 2003, Las Vegas, NV, USA, 8–12 December 2003.
18. Wu, T.D. A Real-World Analysis of Kerberos Password Security. *Ndss*, 3 February 1999 .
19. Kihara, M.; Iriyama, S. New authentication algorithm based on verifiable encryption with digital identity. *Cryptography* **2019**, *3*, 19. [CrossRef]
20. Grassi, P.A.; Garcia, M.E.; Fenton, J.L. *Digital Identity Guidelines*; NIST Special Publication (NIST SP)-800-63-3; National Institute of Standards and Technology: Los Altos, CA, USA, 2017.
21. Accardi, L.; Regoli, M.; Ohya, M. The QP-DYN algorithms. In *QP–PQ Quantum Probability and White Noise Analysis*; Accardi, L., Freudenberg, W., Ohya, M., Eds.; World Scientific: Singapore, 2010; pp. 1–16.
22. Iriyama, S.; Tanaka, Y.; Hara, Toshihide,; Ohya, M. On a PRNG based on non-commutative algebra and its applications. *SPT* **2014**, *13*, 1–3.
23. Proverif Webcite. Available online: https://prosecco.gforge.inria.fr/personal/bblanche/proverif/ (accessed on 2 June 2020).
24. Security Assertion Markup Language (SAML) V2.0 Technical Overview, OASIS, 2008. OASIS. Available online: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html (accessed on 14 May 2020).
25. Iriyama, S.; Kihara, M. ENCRYPTED DATA PROCESSING SYSTEM AND PROGRAM. Japan Patent PCT/JP2018/045505, 27 June 2019.
26. Iriyama, S.; Kihara, M. AUTHENTICATION SYSTEM AND PROGRAM. Japan Patent JP2020/25659, 18 February 2020.