Extended Inter-Device Digital Rights Sharing and Transfer Based on Device-Owner Equality Verification Using Homomorphic Encryption

Yoshihiko OMORI†(a) and Takao YAMASHITA†, Members

SUMMARY In this paper, we propose homomorphic encryption based device owner equality verification (HE-DOEV), a new method to verify whether the owners of two devices are the same. The proposed method is expected to be used for credential sharing among devices owned by the same user. Credential sharing is essential to improve the usability of devices with hardware-assisted trusted environments, such as a secure element (SE) and a trusted execution environment (TEE), for securely storing credentials such as private keys. In the HE-DOEV method, we assume that the owner of every device is associated with a public key infrastructure (PKI) certificate issued by an identity provider (IdP), where a PKI certificate is used to authenticate the owner of a device. In the HE-DOEV method, device owner equality is collaboratively verified by user devices and IdPs that issue PKI certificates to them. The HE-DOEV method verifies device owner equality under the condition where multiple IdPs can issue PKI certificates to user devices. In addition, it can verify the equality of device owners without disclosing to others any privacy-related information such as personally identifiable information and long-lived identifiers managed by an entity. The disclosure of privacy-related information is eliminated by using homomorphic encryption. We evaluated the processing performance of a server needed for an IdP in the HE-DOEV method. The evaluation showed that the HE-DOEV method can provide a DOEV service for 100 million users by using a small-scale system in terms of the number of servers.

key words: authentication, authorization, Internet of things (IoT), PKI, digital rights sharing, digital rights transfer, FIDO, biometrics authentication

1. Introduction

Many popular Internet services, such as social networking services (SNSs), e-commerce, and video-on-demand (VoD), are personalized for each user. To use such services, users need to register accounts. Users consequently have rights to use their own accounts on a server, but they can use their account only when a server succeeds in verifying their asserted identities using authentication methods.

Authentication methods can be categorized into three types in terms of what is used for verifying identities asserted by users: “what you know” (WYK), “what you have” (WYH), and “what you are” (WYA) [1]. WYK, WYH, and WYA are also called “something you know,” “something you have,” and “something you are,” respectively.

WYK authentication methods prove an asserted user’s identity by using secret information that only a proper user knows. Examples include password-based authentication, which is widely used for a variety of applications. WYH authentication methods prove an asserted user’s identity by using a token owned by a proper user. Examples include smartcard-based authentication. WYA authentication methods prove asserted users’ identities by using their biological properties on the basis of physical characteristics of the human body [1]. A very common example is a fingerprint, and numerous smartphones are now equipped with fingerprint-based authentication functions. On the Internet, WYK and WYH authentication methods are widely used for remote authentication, in which a user’s identity is authenticated over communications networks by a server.

Fast Identity Online (FIDO) Alliance [2] has been trying to develop and deploy highly secure and easy-to-use authentication methods. Especially, a universal authentication framework (UAF) [3] standardized by FIDO Alliance improves both security and ease-to-use in an authentication process. In the UAF, WYH and WYA authentication methods are used for remote and local authentication, respectively, where local authentication is an authentication process in which a device such as a smartphone verifies a user’s identity. When a local authentication process invoked by a user is successful, a remote authentication process is automatically executed without any additional user action. From the viewpoint of user experience, a user needs to pay attention only to a local authentication process. A user does not need to know a complicated remote authentication process. From the viewpoint of security, the UAF enables the use of public-key cryptography. This improves the security of remote authentication. In addition, by combining of WYH and WYA authentication methods, the UAF can reduce the risk of unauthorized access to a service account from a stolen device.

In the UAF, remote authentication is secured by public-key cryptography. FIDO Alliance recommends private keys in public-key cryptography to be stored in a secure element (SE)/trusted execution environment (TEE) [4] to prevent them from being improperly used by others. On the other hand, users tend to own and use multiple devices such as a smartphone, tablet computer, personal computer (PC), and other smart devices (e.g., smart watches and glasses). Such users choose the most convenient device depending on their situations. Here, we call a situation where each user
owns and uses multiple devices a multi-device environment. In multi-device environments, WYH authentication methods decrease the usability of services that require authentication because each private key is stored in the SE/TEE of one user device and consequently cannot be used from any other device. Credential sharing is important especially for services that currently protect users’ accounts using WYK authentication methods, such as passwords, and are frequently used in daily life. Examples are SNSs and e-commerce.

To improve usability in multi-device environments for services described in the previous paragraph (e.g., SNSs and e-commerce), we have proposed inter-device credential sharing methods based on the verification of device owner equality [5]–[7], where a credential is secret information to authenticate a user such as a password and a private key. In these methods, we assume that a public key infrastructure (PKI) certificate is issued by an identity provider (IdP) and installed to every user device. A PKI certificate and its corresponding private key are stored in the SE/TEE of each device. Inter-device credential sharing between devices is enabled when the equality of their owners is verified using PKI certificates installed in devices, where we call a process to verify whether the owner of two devices is equal device owner equality verification (DOEV). When a PKI certificate is issued to a user device, its vendor identifies who its owner is in accordance with rules defined by an IdP and then requests an IdP to issue a PKI certificate. Examples of rules for identifying users are defined by standards organizations such as the National Institute of Standards and Technology (NIST) [9]. One example in the rules requires that a user has to be checked through face-to-face identification using one or more of his/her identification (ID) cards.

We previously proposed a DOEV method that satisfies two requirements: R-1) allowing multiple IdPs that issue PKI certificates to user devices and R-2) eliminating disclosure of personally identifiable information (PII) to user devices [10]. When PKI certificates are issued to user devices by a single IdP, DOEV can be easily achieved because the contents of the subject fields of PKI certificates issued to a device owner can be the same, where the subject field of a PKI certificate represents an entity to which it is issued. On the other hand, for users’ convenience, a PKI certificate for a device should be issued when a user obtains it from its vendor. In this situation, we need to assume that multiple IdPs issue PKI certificates to user devices because numerous vendors provide user devices and choose the best IdP for their business. Therefore, requirement R-1) should be satisfied. When different IdPs issue PKI certificates to multiple devices owned by a single user, the contents of the subject fields of these PKI certificates are not generally the same.

When two devices execute a DOEV process, their owners might be different. In addition, IdPs that issue PKI certificates to these devices might be different. In such a case, information to identify the devices owners needs to be exchanged. If we use PII or long-lived IDs to identify users in a DOEV process, it can cause personal information of users to be revealed because various information generated through the behavior of a user can be associated with PII and/or long-lived IDs [11], [12]. Due to the risk of revealing personal information, requirement R-2) must be satisfied. This means that requirement R-2) is caused by allowing multiple IdPs that issue PKI certificates to user devices, which is requirement R-1). The previously proposed DOEV method satisfies requirement R-2) by assigning the role of PII comparison of user devices to IdPs that issue PKI certificates. By assigning this role, we restrict entities sharing PII to only mutually trusted IdPs. We described the motivation for requirements R-1) and R-2) in detail in the previous paper [10], where the original DOEV method is proposed.

On the other hand, management and usage of PII including long-lived IDs are regulated by laws, which differ among countries. Therefore, the new DOEV method has an additional requirement: R-3) the disclosure of PII to other entities in DOEV should be minimized so that a DOEV method can be used internationally. In this paper, we propose a new DOEV method that satisfies requirement R-3) in addition to R-1) and R-2). The newly proposed DOEV method is accomplished using homomorphic encryption to compare PII regarding the owners of two devices. We call this DOEV method a homomorphic encryption based DOEV (HE-DOEV) method.

The remainder of this paper is organized as follows. In Sect. 2, we explain the FIDO UAF and an inter-device credential sharing method that we previously proposed. In Sect. 3, we describe the HE-DOEV method. In Sect. 4, we evaluate the proposed method in terms of its resistance to security threats and the processing performance of a server to provide a DOEV service using the HE-DOEV method. In Sect. 5, we compare our method with related work. In Sect. 6, we conclude the paper.

2. Inter-Device Credential Sharing

2.1 FIDO-Based Authentication Method

As described in Sect. 1, we previously proposed a DOEV method [10] as well as credential sharing methods [5]–[7]. These methods are designed so that they work with the UAF standardized by FIDO Alliance. In this subsection, we briefly introduce the FIDO UAF. Table 1 lists the major abbreviations used in this paper.

The architecture of the FIDO UAF is composed of three types of entities as shown in Fig. 1: a user device, an authentication, and a metadata server. Part of an authentication function using credentials in a user device is implemented in a module called a FIDO authenticator or an authenticator for short. An authenticator is accessed by a client application through an interface module called an authenticator specific module (ASM).

A FIDO authenticator has the functions of signing, signature verification, encryption, and decryption that are needed for user authentication. A FIDO authenticator manages public/private key pairs used for user authentication. These public/private key pairs are usually stored in the
Table 1: Major abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAID</td>
<td>Authenticator Attestation Identifier</td>
</tr>
<tr>
<td>BVR</td>
<td>Bidirectional Verification Request</td>
</tr>
<tr>
<td>DOC</td>
<td>Device Owner Certificate</td>
</tr>
<tr>
<td>DOEV</td>
<td>Device Owner Equality Verification</td>
</tr>
<tr>
<td>HE-DOEV</td>
<td>Homomorphic Encryption based Device Owner Equality Verification</td>
</tr>
<tr>
<td>HPE1</td>
<td>Homomorphic Private Equality Test</td>
</tr>
<tr>
<td>HSM</td>
<td>Hardware Security Module</td>
</tr>
<tr>
<td>PCR</td>
<td>PII Comparison Result</td>
</tr>
<tr>
<td>PIV</td>
<td>Personal Identity Verification</td>
</tr>
<tr>
<td>SOHD</td>
<td>Start of HE-DOEV</td>
</tr>
<tr>
<td>UAF</td>
<td>Universal Authentication Framework</td>
</tr>
<tr>
<td>UVR</td>
<td>Unidirectional Verification Request</td>
</tr>
</tbody>
</table>

SE/TEE of a FIDO authenticator. A public/private key pair is first generated in a FIDO authenticator. The private key of a key pair is never retrieved from it. In addition to signing, signature verification, encryption, and decryption, a FIDO authenticator also has the function of local authentication using biometric information, which is also stored in an SE/TEE.

In the first interaction between a FIDO authenticator and a FIDO server, the FIDO authenticator first generates a public/private key pair and next registers a public key to the FIDO server. The FIDO server authenticates a user using a registered public key after its registration.

A FIDO authenticator product is identified and certified on a device model level from the viewpoint of its capability. When a FIDO authenticator is certified, an authenticator attestation identifier (AAID) is assigned to it. In addition, an attestation key is associated with it. An attestation key is composed of an attestation certificate including a public key and a corresponding private key. The attestation certificate and the private key of an attestation key are stored in a metadata server and a FIDO authenticator, respectively.

When a FIDO authenticator registers a public key used for user authentication to a FIDO server, it sends a signature calculated using the private key of an attestation key assigned to itself. A FIDO server can obtain information about the capability of a FIDO authenticator from a metadata server, where an authenticator is identified using its AAID. A FIDO server can consequently verify the capability of a FIDO authenticator when a public key for a user is registered. If a FIDO authenticator has insufficient capability for a service, a FIDO server can reject the registration of a user’s public key.

2.2 Credential Sharing Methods

To accomplish inter-device credential sharing in multi-device environments, we have proposed two methods: a networked-key-store and a secure-copy based credential sharing method[5]–[7]. Both methods are enabled only when a DOEV process determines that the owners of two devices are equal. We here explain the secure-copy based credential sharing method, which adopts a decentralized approach. In this method, we assume that a PKI certificate is issued by an IdP and installed in the SE/TEE of every user device for identifying its owner. We call this PKI certificate a device owner certificate (DOC) in this paper. Since an IdP issues DOCs to user devices, it must offer a high level of identity assurance as the root of trust, which is quite costly due to the necessity of conducting many verification processes[9]. One example of a user verification process is an in-person verification process, where a user is verified by checking his/her identity (ID) cards in face-to-face communication. Another example is a verification process using an ID card with a smartcard function, which provides signature generation and verification using public-key cryptography. We believe it is reasonable for a business that already confirms people’s identities in their normal operations to also play the role of an IdP issuing DOCs. In Japan, for example, mobile network operators are required by law to confirm the identity of a subscriber before selling him/her a phone and providing access to the mobile network. Users do not have to do any additional action for identity proofing if the mobile operator issues DOCs when contracting with users. The other most economical IdP candidate to issue DOCs is a nationwide PKI system. Some governments already provide a service of issuing certificates to mobile devices[13]. Also, businesses that already have contact points throughout the nation and usually confirm the identities of their customers (e.g., banks or postal services) are expected to cost-effectively act as an IdP issuing DOCs.

Figure 2 shows the secure-copy based credential sharing method. In this method, the copies of credentials in one device are made in another device when the owners of these
devices are verified to be equal using their DOCs. In addition, the copies of credentials in one device can be made in multiple devices by repeating the execution of this method. We have investigated a method to copy credentials between two devices and verified its feasibility by implementing its core part in a commercially available smartphone with a TEE function [7], [8]. This credential copying function is accomplished by using two applications in a user device: a key copy application running on a normal operating system (OS) and a key-coping TEE application (TA). These applications in device $x$ in Fig. 2 first gather data items that are stored in FIDO-related modules, such as an authenticator and an ASM, and are needed for a FIDO authentication process such as authentication keys and as information to identify authentication keys. Then, gathered data items are encrypted by the key-copying TA using a public key in the DOC of device $y$ in Fig. 2 and sent to device $y$ through the key copy application. Prior to sending them, the key-copying TA of device $x$ confirms that of device $y$ by using the attestation key of device $y$ to prevent private keys from being transferred to an untrusted area.

The method described just above accomplishes credential sharing without changing the registration and authentication protocols of the FIDO UAF. On the other hand, the current specifications of the FIDO UAF do not assume that a private key and its corresponding data items needed for a FIDO authentication process are transferred from one device to another. Therefore, the usage of such data items should be clarified in the FIDO UAF specifications in order to widely deploy credentials sharing.

3. Homomorphic Encryption Based Device Owner Equality Verification

3.1 Requirements

The credential sharing methods described in the previous section enable the credentials of a user to be shared among multiple devices. These methods need DOEV as described in Sect. 2.2. We assume that DOCs in the SE/TEE of user devices can be used for DOEV. As described in Sect. 1, a DOEV method needs to satisfy three requirements:

R-1) allowing multiple IdPs that issue PKI certificates to user devices,

R-2) eliminating the disclosure of the PII and long-lived IDs of a user device to other devices for privacy, and

R-3) eliminating the disclosure of PII and long-lived IDs managed by an entity to any other IdP in addition to any other device, which corresponds to R-2).

Requirements R-1) and R-2) are the same as those of the previously proposed DOEV method, whereas requirement R-3) is newly introduced in this paper. Requirements R-2) and R-3) mean that the disclosure of PII and long-lived IDs has to be eliminated for protecting users’ privacy. Requirements R-2) and R-3) are caused by R-1) because we have to compare the PII and/or long-lived IDs of user devices managed by different entities under the condition where requirement R-1) is satisfied.

3.2 Assumptions

In this subsection, we describe assumptions for the HE-DOEV method using Fig. 3(a). We need to allow multiple IdPs that issue DOCs from requirement R-1). Therefore, there are two user devices $x$ and $y$ and two IdPs $a$ and $b$ in the figure, where IdPs $a$ and $b$ issue the DOCs of devices $x$ and $y$, respectively. We call device $x$ ($y$) the counter device of device $y$ ($x$). When a DOC is issued to a user device, its vendor identifies who its owner is by using his/her ID cards. To satisfy requirement R-1), we need to compare PII and/or long-lived IDs contained in ID cards. Therefore, we assume that an IdP manages the PII and long-lived IDs of a device owner after it has confirmed them. From requirements R-2) and R-3), PII and/or long-lived IDs associated with device owners must be compared without disclosing PII and long-lived IDs to other entities. In the issuance of DOCs, a public/private key pair for a DOC is generated in the SE/TEE of a user device. A distinguished name in the subject field of a DOC is generally determined by a certificate authority (CA) operated by an IdP. Therefore, when different IdPs issue DOCs for devices owned by the same owner, the distinguished names of these DOCs are generally different.

3.3 Mutual Identification between User Devices

In an HE-DOEV process, devices $x$ and $y$ in Fig. 3(a) have to identify each other. On the other hand, we cannot use PII and long-lived IDs to identify the owner of a device due to requirement R-2) as described in Sect. 3.1. Therefore, we cannot use a DOC to identify the owner of a device because it includes an ID in the subject field of a certificate, which are used for a long time. To eliminate the use of PII and long-lived IDs, we introduce an additional and temporary PKI certificate, which is called a pseudonym certificate, for a user device in the HE-DOEV method. A pseudonym certificate is issued
to a user device by the same IdP as the issuer of its DOC. A pseudonym certificate is used only in the HE-DOEV method for a short time, and its subject field contains random strings. In addition, the subject field of a pseudonym certificate for a user device is changed every time it is issued and updated so that a user device becomes difficult to track.

3.4 PII Arrangement

As described in the previous subsection, user devices use pseudonym certificates to identify each other. However, a user device cannot verify whether its owner is equal to the other device’s owner using pseudonym certificates because they generally have different IDs associated with the same owner and do not include PII. Therefore, we have to find the PII data items of device owners associated with pseudonym certificates and compare them. The PII of a user device can be managed by a user device and/or an IdP that issues a DOC to a user device. As a result, we have three alternatives to compare the PII of user devices: (A) device-to-device, (B) device-to-IdP, and (C) IdP-to-IdP comparisons as shown in Fig. 3 (b), where we assume that a PII data item in a user device is stored in its SE/TEE by an IdP so that its PII data item cannot be modified even by its owner. From requirements R-2) and R-3), no user devices or IdPs can disclose the PII data items of user devices. To eliminate the disclosure of PII data items, we use a homomorphic private equality test (HPET) protocol [14] for the HE-DOEV method, where the objective of the HPET protocol is to compare information without leaking it. We describe the HPET protocol in the next subsection.

When we use the HPET protocol [14] to compare the PII data items of owners of user devices, any of comparisons (A), (B), and (C) can be used because the HPET protocol can compare information without leaking it. On the other hand, small mobile devices like smartphones tend to be lost more often than large and/or non-mobile devices such as desktop PCs. When we lose our smartphones, we can request a network service provider to suspend the service for the lost smartphones. In addition to a network service, a service using the HE-DOEV method should be also designed so that it can be suspended at a user’s request. To enable a service to be suspended by using the HE-DOEV method when a user has lost his/her device, an IdP needs to be incorporated in the process of the HE-DOEV method. Hence, comparisons (B) and (C) can be used. As described above, we use the HPET protocol for the HE-DOEV method. The HPET protocol contains two entities: a chooser and a sender, where they hold information that is compared. The HPET protocol generally imposes a heavy load on both the chooser and sender. When we select comparison (C), a user device should be notified of the result of the HPET protocol by the most trustworthy IdP that issues a DOC to it. To achieve this, each IdP has to play the roles of both a chooser and sender in the HPET protocol in order to know its result, because only a chooser can know the result of the HPET protocol. On the other hand, when we select comparison (B), a user device has to take the role of a chooser, because a user device has to know the result of the HPET protocol for inter-device credential sharing. As a result, the role of an IdP is only that of a sender when we use comparison (B). This leads to the HPET protocol imposing a lower server load than in comparison (C). Hence, we adopt comparison (B) for the HE-DOEV method.

When we adopt comparison (B), a user device needs to execute the HPET protocol with an IdP, where devices identify each other using a user device’s pseudonym and an IdP’s certificate. In addition, user devices participating in an HE-DOEV process identify each other using their pseudonym certificates. Hence, we assume that user devices and IdPs hold the list of IdPs that can be trusted. This list of trusted IdPs held by user devices and IdPs has to be adequately maintained for security. We describe how to manage the list of trusted IdPs in a user device in Sect. 3.7. On the other hand, how to manage the list of trusted IdPs held by IdPs is out of the scope of this paper because it needs further investigation.

3.5 PII Equality Test Using Homomorphic Encryption

We use the HPET protocol [14] to compare the PII data items of user devices in the HE-DOEV method as described in the previous subsection. In this subsection, we first introduce a homomorphic encryption scheme and the HPET protocol that is achieved using a homomorphic encryption scheme over an additive group. We then explain how to verify the equality of PII data items using the HPET protocol.

An example of homomorphic encryption schemes over an additive group is a Paillier cryptosystem [15]. In the Paillier cryptosystem, its encryption function using public key pk $E_{pk}$ has the following property:

$$E_{pk}(m_1; r_1) \cdot E_{pk}(m_2; r_2) = E_{pk}(m_1 + m_2; r_1 \cdot r_2),$$

(1)

where $m_1$, $r_1$, and $E_{pk}(m_1; r_1)$ are a message, a random value, and a function to encrypt data $m_1$ using public key pk, respectively. $E_{pk}(m_1; r_1)$ in the above formula is represented as

$$E_{pk}(m; r_i) = (1 + N)^m \cdot r_i^N \mod N^2,$$

(2)

where $N$ is a product of two distinct primes $p$ and $q$ of equal length (cf. $N = pq$), and message $m$ and random value $r_i$ are included in $Z_N$. By using the property represented by formula (1), we can calculate the sum of two messages without decrypting them [14], [16].

We next explain the HPET protocol, which uses a homomorphic encryption scheme. The HPET protocol contains two entities: a chooser and a sender. The chooser and sender have secret data items $V_c$ and $V_s$, respectively. The chooser and sender cooperate, and only the former can know whether $V_c$ and $V_s$ are equal. To compare information without leaking it, the difference between $V_c$ and $V_s$ is calculated in an encrypted form. We describe the HPET protocol below.
1. A chooser generates a new key pair \((pk, sk)\) and calculates \(c = E_{pk}(V_x; r)\), where \(r\), \(pk\), and \(sk\) are a random value, a public key, and a private key in \(Z^*_p\), respectively. He then sends \(E_{pk}(V_x; r)\) and public key \(pk\) to a sender. We call this message sent to a sender a chooser message.

2. A sender generates random values \(s\) and \(r'\) in \(Z^*_p\) and calculates \(c' = (c \cdot E_{pk}(-V_y; 1))^{r'} \cdot E_{pk}(0; r')\). He then sends \(c'\) to a chooser. Here, \(c'\) is apparently equal to \(E_{pk}(s(V_x - V_y); r' \cdot r')\) from formula (1). We call \(c'\) sent to a chooser a sender message.

3. A chooser accepts that \(V_x\) is equal to \(V_y\) if and only if \(D_{sk}(c')\) is equal to zero, where \(D_{sk}(c')\) is a function to decrypt cipher text \(c'\) using private key \(sk\).

As described in Sect. 3.4, we adopt device-to-IdP comparison for verifying the equality of PII data items of user devices. Therefore, the PII data items of a user device are placed in the device itself and an IdP that issues its DOC. In the example in Fig. 3 (a), the PII data items of devices \(x\) and \(y\) are placed in IdPs \(a\) and \(b\), respectively, in addition to the devices themselves. As we adopt device-to-IdP comparison, the PII data items of device \(x\) \((y)\) are compared with those of \(y\) \((x)\) placed in IdP \(b\) \((a)\) in Fig. 3 (a). In addition, a user device and an IdP play those roles of a chooser and a sender in the HPET protocol, respectively, because only a chooser can know the result of the HPET protocol, and a user device has to know it for inter-device credential sharing.

Let \(V_x\) and \(V_y\) be the PII data items of devices \(x\) and \(y\), respectively. To verify the equality of the PII data items of devices \(x\) and \(y\), we have to compare \(V_x\) and \(V_y\). \(V_x\) is placed in both device \(x\) and IdP \(a\). Similarly, \(V_y\) is placed in both device \(y\) and IdP \(b\).

In the HPET protocol, a chooser first calculates \(c\). Let \(c_x\) and \(c_y\) be \(E_{pk}(V_x; r_x)\) and \(E_{pk}(V_y; r_y)\), where \(pk_x\) and \(sk_x\), \(pk_y\) and \(sk_y\), and \(r_x\) and \(r_y\) are a public and a private key generated by \(x\), those generated by \(y\), and random values generated by devices \(x\) and \(y\), respectively. \(c_x\) and \(c_y\) are calculated by devices \(x\) and \(y\) to compare \(V_x\) and \(V_y\) with \(V_x\) and \(V_y\) in IdPs \(b\) and \(a\), respectively. \(c_x\) and \(c_y\) are transmitted from devices \(x\) and \(y\) to IdPs \(b\) and \(a\), respectively.

After \(c\) is calculated in the HPET protocol, a sender calculates \(c'\). Let \(c'_x\) and \(c'_y\) be \((c_x \cdot E_{pk}(-V_x; 1))^{r'_x} \cdot E_{pk}(0; r'_x)\) and \((c_y \cdot E_{pk}(-V_y; 1))^{r'_y} \cdot E_{pk}(0; r'_y)\), where \(s_x\) and \(r'_x\), and \(s_y\) and \(r'_y\) are random values generated by IdPs \(a\) and \(b\), respectively. \(c'_x\) and \(c'_y\) are transmitted from IdPs \(a\) and \(b\) to devices \(y\) and \(x\), respectively.

When devices \(x\) and \(y\) receive \(c'_x\) and \(c'_y\), they calculate \(D_{sk}(c'_x)\) and \(D_{sk}(c'_y)\), respectively. Devices \(x\) and \(y\) accept \(V_x = V_y\) if and only if \(D_{sk}(c'_x)\) and \(D_{sk}(c'_y)\) are equal to zero, respectively.

3.6 Overview of HE-DOEV Method

In the HE-DOEV method, the PII data items of two users stored in user devices and IdPs are compared. The PII data items managed by an IdP are originally registered by users.

Therefore, when an IdP uses the PII data items of a user, it needs to verify whether the owner of a user device accepts the use of his/her PII data items for an HE-DOEV process. In addition, a user device is identified by IdPs and other devices using a pseudonym certificate as described in Sect. 3.3. Therefore, the HE-DOEV method is composed of four procedures:

1. the issuance of a pseudonym certificate,
2. agreement on HE-DOEV process execution,
3. the verification of agreement on HE-DOEV process execution, and
4. the comparison of PII data items of devices.

Figure 4 shows the above procedures in the HE-DOEV method. This figure depicts two execution environments in a device: a rich OS and an SE/TEE. Processes performed by a device in an HE-DOEV process are realized by applications that run in these environments. We call applications running in an IdP and an SE/TEE an IdP server and a secure area application (AP), respectively. The HE-DOEV method is designed on the basis of an assumption about a security threat to the method: malicious users never falsify software on an IdP server and a secure-area AP.

In the explanation of the above four procedures, we use Figs. 5 and 6. To simplify Fig. 6, we omit the message sequences of transport layer security (TLS) [17]. In addition, we only explain the processes of one user device in Fig. 4, which is device \(x\), because the processes of user devices \(x\) and \(y\) are symmetric. Table 2 lists major symbols used in Sect. 3.6 and the following sections.

3.7 Issuance of Pseudonym Certificate

Prior to the execution of the HE-DOEV method, devices \(x\) and \(y\) establish a peer-to-peer communication channel, which is depicted as (0) in Fig. 6. We can use a near-field communication technology such as Bluetooth for a user’s easy pairing. In the following explanation, we assume that this established peer-to-peer communication channel is used when devices \(x\) and \(y\) communicate directly.

A pseudonym certificate is used by an IdP and a user...
OMORI and YAMASHITA: EXTENDED INTER-DEVICE DIGITAL RIGHTS SHARING AND TRANSFER USING HOMOMORPHIC ENCRYPTION

Fig. 6 Message sequence of proposed HE-DOEV method

(1) - Procedure for device x to verify device-owners’ equality (explained in Sect. 3)
(2) - Procedure for device y to verify device-owners’ equality (explanation omitted in Sect. 3)
(3) - Pre-procedure for HE-DOEV

Table 2 Symbols used for describing proposed HE-DOEV method

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_d^{(o)} )</td>
<td>Device owner certificate of device d</td>
</tr>
<tr>
<td>( f_d^{(p)} )</td>
<td>Pseudonym certificate of device d</td>
</tr>
<tr>
<td>( f_i^{(p)} )</td>
<td>Certificate of IdP i</td>
</tr>
<tr>
<td>( N_{j,d} )</td>
<td>Nonce sent to device d by IdP i</td>
</tr>
<tr>
<td>( V \parallel W )</td>
<td>Data created by concatenating data V and W</td>
</tr>
<tr>
<td>( m \parallel \text{sig}(m, K) )</td>
<td>Signature of message m with key K</td>
</tr>
<tr>
<td>( M_d(m, K_1, K_2) )</td>
<td>Message to be signed</td>
</tr>
<tr>
<td>( Q_d )</td>
<td>Request message of device d for issuance of pseudonym certificate</td>
</tr>
<tr>
<td>( S_d^{(p)} )</td>
<td>Private key corresponding to device owner certificate of device d</td>
</tr>
<tr>
<td>( S_d^{(p)} )</td>
<td>Private key corresponding to pseudonym certificate of device d</td>
</tr>
<tr>
<td>( S_d^{(c)} )</td>
<td>Attestation key (private key) assigned to secure area application in device d</td>
</tr>
<tr>
<td>( A_d )</td>
<td>Authenticator attestation ID (AAID) of device d</td>
</tr>
<tr>
<td>( E_{pk} )</td>
<td>Encryption function in homomorphic encryption scheme, where public key is pk</td>
</tr>
<tr>
<td>( D_{sk} )</td>
<td>Decryption function in homomorphic encryption scheme, where private key is sk</td>
</tr>
<tr>
<td>( c_d )</td>
<td>Chooser message created by device d in HPET protocol</td>
</tr>
<tr>
<td>( c_d' )</td>
<td>Sender message corresponding to ( c_d ) in HPET protocol</td>
</tr>
<tr>
<td>( pk_d )</td>
<td>Public key generated by device d for HPET protocol</td>
</tr>
<tr>
<td>( sk_d )</td>
<td>Private key generated by device d for HPET protocol</td>
</tr>
</tbody>
</table>
IdP $a$ and a secure-area AP in device $x$ then authenticate each other, where we assume that they are never altered by malicious users, as described in Sect. 3.6. The secure-area AP of device $x$ second generates a public/private key pair in its SE/TEE and creates a certificate request message in accordance with the specification of PKCS#10 [18], where this request message is signed with a private key generated by the secure-area AP of device $x$. The certificate request message of device $x$ for its pseudonym certificate is further signed with two private keys $S_{x}^{(o)}$ and $S_{x}^{(A)}$, where $S_{x}^{(o)}$ and $S_{x}^{(A)}$ are private keys corresponding to the DOC and the attestation key of device $x$, respectively. Signing with these private keys is performed in this order. The former signature is needed for associating a certificate request with a DOC. The latter one is needed to prove that a certificate request message is created in the secure-area AP of a device. As described in the following subsections, messages sent from a user device in the HE-DOEV method are signed with the private key corresponding to its attestation key for the same reason. Here, we assume that a function to calculate a signature with the private key of an attestation key is not provided to applications running on a rich OS. This function can be used by only a secure-area AP. As a result, other devices and IdPs can verify whether a certificate request is created by a secure-area AP.

The request message of device $x$ for issuing a pseudonym certificate is sent to IdP $a$ ((1)-1 in Fig. 6). Let $I_{x}^{(o)}, Q_{x},$ and $S_{x}^{(p)}$ be the DOC of device $x$, a certificate request created by device $x$, and a private key corresponding to a public key in $Q_{x}$, respectively. Then, a certificate request message sent to IdP $a$ by device $x$ is $M_{x}(M_{x}(Q_{x}, S_{x}^{(p)})) || I_{x}^{(o)} || A_{x}, S_{x}^{(o)}, S_{x}^{(A)}$, where $A_{x}, M_{x}(m, K), M_{x}(m, K_{1}, K_{2}),$ and $m_{1} || m_{2}$ are the AAID of device $x$, a message that consists of message $m$ and a signature for $m$ calculated with key $K, M_{x}(m, K_{1}, K_{2}),$ and a message created by concatenating data items $m_{1}$ and $m_{2}$, respectively. $I_{x}^{(o)}$ and $A_{x}$ are needed for verifying signatures calculated with $S_{x}^{(o)}$ and $S_{x}^{(A)}$, respectively.

When IdP $a$ receives a certificate request, three signatures associated with it are first verified. IdP $a$ then issues and sends a pseudonym certificate $I_{a}^{(p)}$ to device $x$ ((1)-2 in Fig. 6). The message of IdP $a$ to send a pseudonym certificate to device $x$ is $M_{a}(I_{a}^{(p)} || I_{a}^{(c)}, S_{a}^{(c)})$, where $I_{a}^{(c)}$ and $S_{a}^{(c)}$ are the PKI certificate of IdP $a$ and a private key corresponding to $I_{a}^{(c)}$. $I_{a}^{(c)}$ is needed for verifying a signature calculated with $S_{a}^{(c)}$. Below, we list the messages exchanged between IdP $a$ and device $x$ to issue a pseudonym certificate, which are described above. (1)-1 and (1)-2 of the beginning parts of the following lines represent the numbers of processes shown in Fig. 6.

\begin{align*}
(1)-1: & \quad M_{x}(M_{x}(Q_{x}, S_{x}^{(p)})) || I_{x}^{(o)} || A_{x}, S_{x}^{(o)}, S_{x}^{(A)} \\
(1)-2: & \quad M_{a}(I_{a}^{(p)} || I_{a}^{(c)}, S_{a}^{(c)}) \\
\end{align*}

When IdP $a$ issues $I_{a}^{(p)}$ to device $x$, it sends the list of IdPs trusted by IdP $a$ along with $I_{a}^{(p)}$. This list is signed by IdP $a$. Device $x$ uses this list to determine whether IdP $b$ is trustable in the proposed method.

### 3.8 Agreement on HE-DOEV Process Execution

We first describe the overview of an agreement on HE-DOEV process execution. As described in Sect. 3.4, we adopt device-to-IdP comparison to verify the equality of the PII data items of devices $x$ and $y$. In addition, we use the HPET protocol for this verification as described in Sect. 3.5. When using the HPET protocol, a device and an IdP are a chooser and a sender, respectively, because a device has to know the result of the HPET protocol to determine whether it can execute credential sharing. When an IdP participates in an HE-DOEV process, it has to verify whether both the owners of two devices intend to execute an HE-DOEV process because an IdP plays the role of a sender by using the PII of a user device. For example, IdP $a$ in Fig. 5 has to verify whether the owners of devices $x$ and $y$ intend to execute an HE-DOEV process.

Prior to devices $x$ and $y$ agreeing on the execution of an HE-DOEV process, they have to identify each other. They achieve this by using their pseudonym certificates and corresponding private keys. Devices $x$ and $y$ then agree on the execution of an HE-DOEV process. The procedure of this agreement consists of two steps. In the first step, the owner of device $y$ creates a message called a unidirectional verification request (UVR) and sends it to device $x$. This message represents that the owner of device $y$ intends to execute an HE-DOEV process with device $x$. In the second step, when device $x$ receives a UVR message from device $y$ in the first step, it creates a message that represents the approval of the received UVR message. We call this message a bidirectional verification request (BVR). A BVR message created by device $x$ is sent to IdP $a$ for the next procedure in the HE-DOEV method. A BVR message represents that the owners of devices $x$ and $y$ approve the execution of an HE-DOEV process.

As described above, a user device and an IdP play the roles of a chooser and a sender, respectively, in the HPET protocol. In the HPET protocol, a chooser first sends a message, which is the chooser message described in Sect. 3.5, to a sender. We let $c_{y} = E_{pk}(V_{y}, r_{y})$ be a chooser message sent by device $y$. When device $y$ executes an HE-DOEV process with device $x$, device $y$ and IdP $a$ are a chooser and a sender, respectively, in the HPET protocol. Therefore, device $y$ has to transmit $c_{y}$ to IdP $a$. On the other hand, a UVR message sent by device $y$ is transmitted to IdP $a$ through device $x$ as part of a BVR message. To reduce the complexity of messages exchanged in the HE-DOEV method, $c_{y}$ is piggy-backed on a UVR messages created by device $y$.

We next describe the agreement on HE-DOEV process execution in detail using Fig. 5 as well as Fig. 6. Before starting the process of agreement on HE-DOEV process execution, devices $x$ and $y$ first establish a TLS connection using their pseudonym certificates. The secure-area APs of these devices next authenticate each other similarly to the authentication of a user device and an IdP in the issuance of a pseudonym certificate described in Sect. 3.7 ((2)-1 in...
Fig. 6). Then, the processes of user devices x and y described below are performed by their secure-area APs.

After devices x and y identify each other, they agree on the execution of an HE-DOEV process. Device x first sends a message called the start of HE-DOEV (SOHD) message to IdP a ((2)-2 in Fig. 6). An SOHD message notifies an IdP of the start of an HE-DOEV process. This SOHD message includes the pseudonym certificates of devices x and y. An SOHD message sent to IdP a by device x is

\[2\)-2: \( M(x) \parallel p(x) \parallel A_x, S_x, S_x(A) \).

When IdP a receives an SOHD message, it generates nonce values for user devices x and y, where we let \(N_{x,a}\) and \(N_{y,a}\) be nonce values for devices x and y generated by IdP a, respectively. IdP a then sends \(N_{x,a}\) and \(N_{y,a}\) to device x ((2)-3 in Fig. 6). After device x receives \(N_{x,a}\) and \(N_{y,a}\), it forwards \(N_{x,a}\) to device y ((2)-4 in Fig. 6). An IdP uses nonce values to verify whether UVR and BVR messages sent by user devices are present ones. In Fig. 5, the transmissions of nonce values from IdPs are indicated by (2)-3 and (2)-4.

After devices x and y receive nonce values from IdP a, they begin two steps for agreement on the HE-DOEV process execution described in the second paragraph of this subsection. In the first step, device y creates a UVR message and sends it to device x. A UVR message created by device y contains \(I^{(p)}_y, I^{(c)}_y, N_{y,a}, pky, c_y,\) and \(A_y\), where \(pky, c_y,\) and \(A_y\) are a public key generated by device y for the HPET protocol, \(E_{pky}(V_y, r_y),\) and the AAID of device y, respectively. A UVR message is signed with the private keys corresponding to the pseudonym certificate and attestation key of device y in this order. Device y sends this UVR message to device x ((2)-5 in Figs. 5 and 6).

In the second step for agreement on HE-DOEV process execution, when device x receives a UVR message from device y, it creates a BVR message by adding nonce value \(N_{x,a}\) to a received UVR message and by signing with the private keys corresponding to its pseudonym certificate and attestation key in this order. A created BVR message is sent to IdP a for the next procedure in the HE-DOEV method. A BVR message created by device x is sent to IdP a ((2)-6 in Figs. 5 and 6).

A UVR message and a BVR message created by device y and x are

\[2\)-5: \( M_y(I^{(p)}_x, N_{x,a}) || pky || c_x || A_x, S_x, S_x(A) \) and

\[2\)-6: \( M_x(U_{c,y} || N_{x,a} || A_x, S_x, S_x(A) \),

respectively, where \(U_{c,y}\) is a UVR message sent to device x by y.

3.9 Verification of Agreement on HE-DOEV Process Execution

When IdP a receives a BVR message from device x, it first confirms that a received message contains pseudonym certificates the same as those in a received SOHD message (or \(I^{(p)}_x\) and \(I^{(c)}_y\)) and nonce values sent from itself (or \(N_{x,a}\) and \(N_{a,y}\)). IdP a then verifies two signatures associated with a BVR message and two signatures associated with a UVR message included in a BVR message. Through the procedure described just above ((3) in Fig. 6), IdP a verifies that devices x and y agree on the execution of an HE-DOEV process.

3.10 Comparison of PII

When an agreement on HE-DOEV process execution described in the previous subsection is successfully verified, IdP a executes a process to compare the PII data items of devices x and y. As described in Sect. 3.5, we use the HPET protocol to compare PII data items. As a sender in the HPET protocol, IdP a creates a chooser message of device y, which is \(c_y\) included in a received BVR message. IdP a then calculates a sender message \(c'_y\) by using the received \(c_y\).

After calculating \(c'_y\), IdP a creates a message called a PII comparison result (PCR) message. A PCR message contains \(I^{(p)}_x, I^{(c)}_y, I^{(c)}_a,\) and \(c'_y\). The pseudonym certificates of devices x and y \(I^{(p)}_y\) and \(I^{(c)}_y\) in a PCR message are needed to prevent \(c'_y\) from being used for PCR messages of other pairs of devices by malicious users. A PCR message is signed by IdP a. A PCR message sent by IdP a is

\[4\)-1,2: \( M_x(I^{(p)}_y, I^{(c)}_y, c'_y, I^{(c)}_a, S_x(A) ).

A created PCR message is first sent to device x by IdP a ((4)-1 in Fig. 6). Device x then forwards it to device y, which is a chooser in the HPET protocol ((4)-2 in Fig. 6). When IdP a receives the PCR message, it validates the signature of the received PCR message and decrypts \(c'_y\) by calculating \(D_{cy}(c'_y)\). Device y accepts that the owners of devices x and y are the same if \(D_{cy}(c'_y)\) is equal to zero ((4)-3 in Fig. 6).

We have described the HE-DOEV method from Sect. 3.6 to here. This method is based on the previously proposed DOEV method [10]. We next explain the difference between these methods. In the previously proposed DOEV method, we adopt IdP-to-IdP comparison without using homomorphic encryption for the PII of user devices, where PII data items are placed in IdPs. Therefore, the result of PII comparison is notified to a device by an IdP after it exchanges PII with the other IdP. On the other hand, the HE-DOEV method uses device-to-IdP comparison. To achieve this, a chooser message in the HPET protocol is piggybacked on a UVR and a BVR message. Then, a sender message in the HPET protocol calculated by an IdP is notified to a device without any interaction between IdPs. As a result, the PII data items managed by an entity are never disclosed to any other entity in the HE-DOEV method.

3.11 Device Owner Equality Verification for Devices with Certificates Issued by Single IdP

In the HE-DOEV method described above, we assume that two devices participating in an HE-DOEV process have DOCs issued by different IdPs. This HE-DOEV method can
be used under the condition where two devices have DOCs issued by a single IdP. We explain the procedures of the HE-DOEV method for this condition below. In the following explanation, we assume that devices $x$ and $y$ have DOCs issued by IdP $a$.

In the first procedure of the HE-DOEV, which is the issuance of pseudonym certificates described in Sect. 3.7, devices $x$ and $y$ obtain their pseudonym certificates from IdP $a$. In the second procedure of the HE-DOEV, which is agreement on HE-DOEV process execution described in Sect. 3.8, devices $x$ and $y$ receive nonce values from IdP $a$, create UVR and BVR messages as described in Sect. 3.8, and then send BVR messages to IdP $a$. The third procedure of the HE-DOEV, which is the verification of agreement on HE-DOEV method described in Sect. 3.9, is the comparison of PII described in Sect. 3.10, IdP $a$ calculates $c'_x$ and $c'_y$ and sends them to devices $x$ and $y$, respectively. Device $x$ and $y$ can determine if their owners are equal by decrypting $c'_x$ and $c'_y$, respectively.

### 3.12 Cooperation of Inter-Device Credential Sharing and Device Owner Equality Verification

The inter-device credential sharing methods described in Sect. 2.2 cooperate with a DOEV method including the HE-DOEV method proposed in this paper. We briefly explain how the secure-copy based credential sharing method cooperates with the HE-DOEV method.

When device $x$ has credentials used for services and shares them with device $y$ using the secure-copy based credential sharing method, devices $x$ and $y$ first execute the HE-DOEV method. When the owners of devices $x$ and $y$ are equal, the secure-area APs of devices $x$ and $y$ can determine the equality of their owners by decrypting the received $c'_x$ and $c'_y$, respectively. Their secure-area APs then establish a secure channel between them. The secure-area AP of device $x$ encrypts credentials stored in device $x$ (e.g., public/private key pairs) using a public key in the pseudonym certificate of device $y$ and sends them to device $y$. Prior to sending them, the key-copying TA of device $x$ confirms that of device $y$ by using the attestation key of device $y$ to prevent private keys from being transferred to an untrusted area. When device $y$ receives the encrypted credentials from device $x$, it decrypts and stores them in its SE/TEE.

### 4. Evaluation

We evaluated the proposed HE-DOEV method in terms of resistance to security threats and the processing performance of an IdP server. In the former evaluation, we first analyzed security threats to assets in the HE-DOEV method and qualitatively evaluated the resistance of the HE-DOEV method to security threats. In the latter evaluation, we measured the processing performance of an IdP server to estimate the scale of the number of IdP servers to provide a DOEV service using the HE-DOEV method. In this measurement, we focused on two types of processes: processes regarding the HPET protocol and public-key cryptography in the HE-DOEV method. This is because these processes generally impose a heavy load on servers [19], [20] and we aimed at measuring the scale of the number of IdP servers.

#### 4.1 Security

##### 4.1.1 Threat Analysis

When we use the HE-DOEV method proposed in this paper for inter-device credential sharing described in Sect. 3.2, there are two types of assets: credentials used for user authentication and PII data items.

We first analyze security threats to credentials used for user authentication using Fig. 7. As described in Sect. 2, we assume that credentials are stored in the SE/TEE of a user device. In addition, we assume that attackers never falsify software on an IdP server and a secure-area AP. There are two types of assets: credentials used for user authentication and PII data items. We next analyze security threats to PII data items, which are stored in both user devices and IdP servers in the proposed method. We assume that attackers never falsify software on an IdP server and a secure-area AP. There-
fore, we have to examine whether PII data items could be leaked by eavesdropping upon messages exchanged in the HE-DOEV method. We call this scenario Threat 4. We discuss the resistance of the HE-DOEV method to attacks associated with Threats 1 to 4 in the following subsections.

4.1.2 Threat 1: Impersonating Other Devices

In Threat 1, device \( x \) tries to impersonate device \( y \) by obtaining a pseudonym certificate issued to \( y \) and to steal credentials stored in \( y \). In the issuance of a pseudonym certificate, device \( y \) creates a certificate request and sends it to IdP \( b \). A certificate request is signed using three private keys as described in Sect. 3.7. Two of them are \( S_y^{(o)} \) and \( S_y^{(A)} \). In addition, these private keys are stored in the SE/TEE of device \( y \) and cannot be retrieved from it. Therefore, when device \( x \) tries to obtain a pseudonym certificates for device \( y \), it needs signatures calculated using these private keys in order to create a certificate request message. On the other hand, we assume that the software programs of an IdP server and a secure-area AP in a user device are never falsified. In addition, a signing function using the private key of an attestation key can be used only by a secure-area AP. Hence, it is difficult for device \( x \) to obtain the pseudonym certificate of device \( y \). This means that the proposed method is resistant to Threat 1.

4.1.3 Threat 2: Inducing Incorrect Computation in HPET

In Threat 2, device \( x \) induces incorrect computation performed by IdP \( a \) in the HPET protocol by changing a UVR and/or BVR message. To make device \( y \) determine that the owners of devices \( x \) and \( y \) are the same, device \( x \) has to induce IdP \( a \) to send a PCR message with \( c'_x \) such that \( D_{sk}(c'_x) \) is equal to zero. To achieve this, we have two alternatives. One is to replace \( I_s^{(r)} \) in a UVR message sent by device \( y \) with the pseudonym certificate of a user device whose owner is equal to that of device \( y \). The other is to replace \( E_{pk}(V_y; r_y) \| pk_y \) in a UVR message sent by device \( y \) with \( E_{pk}(V_x; r_x) \| pk_x \). However, a UVR message is signed with \( S_y^{(o)} \) and \( S_y^{(A)} \), which are never retrieved from device \( y \). Therefore, a UVR message is difficult to modify. This means that the proposed method is resistant to Threat 2.

4.1.4 Threat 3: Changing Result Computed in HPET

In Threat 3, a PCR message sent by IdP \( a \) is changed so that \( D_{sk}(c'_x) \), where \( c'_x \) is included in a PCR message, is equal to zero. We have two scenarios for it. The first one is to replace \( c'_x \) with \( E_{pk}(0; r) \), where \( r \) is a random number. However, device \( y \) can detect that \( c'_x \) has been modified because a PCR message is signed with \( S_y^{(o)} \).

In the second one, devices \( x, y, \) and \( z \) appear. We assume that device \( z \) tries to steal credentials in device \( y \) by utilizing device \( x \), which has the same owner as device \( y \), and the DOC of device \( z \) is issued by IdP \( a \). When device \( z \) executes an HE-DOEV process with device \( y \), it also requests device \( x \) to execute an HE-DOEV process with device \( y \) at the same time. Device \( x \) then forwards the PCR message received from IdP \( a \) to device \( z \). Device \( z \) finally replaces the PCR message received from IdP \( a \) with that forwarded by device \( x \) and sends it to device \( y \). However, PCR messages sent by IdP \( a \) in HE-DOEV processes between devices \( z \) and \( y \) include \( I_z^{(p)} \) and \( I_y^{(p)} \), and those between devices \( x \) and \( y \) include \( I_x^{(p)} \) and \( I_y^{(p)} \). Therefore, device \( y \) can detect a PCR message has been modified. From the above discussion, the proposed method is resistant to Threat 3.

4.1.5 Threat 4: Leakage of PII

PII data items are stored in the SEs/TEEs of user devices and IdP servers. To compare PII data items, they are exchanged among entities in an encrypted form. A UVR message sent by device \( y \) contains \( E_{pk}(V_y; r_y) \). An entity other than device \( y \) needs private key \( sk_y \) to know \( V_y \). Private key \( sk_y \) can be used only by the secure-area AP of device \( y \). Therefore, no other entity other than device \( y \) can know \( V_y \). A PCR message sent by IdP \( a \) contains \( E_{pk}(V_y; r_y) \) and is equal to \( E_{pk}(s_a(V_y - V_x); r_y^a \| r_y^a) \). Only device \( y \) can know the value of \( s_a(V_y - V_x) \). This value is a random value other than zero when \( V_y - V_x \) is not equal to zero. Therefore, even device \( y \) cannot know \( V_y \) when its owner is not the same as that of device \( x \). As a result, the proposed method is resistant to Threat 4.

4.2 Processing Performance

4.2.1 Measurement Conditions

We explain measurement conditions to evaluate the processing performance of an IdP server using Fig. 4. As described in Sect. 3.6, the HE-DOEV method consists of four procedures: (1) the issuance of pseudonym certificate, (2) agreement on HE-DOEV process execution, (3) the verification of the agreement, and (4) PII comparison. In these procedures, processes regarding public-key cryptography and the HPET protocol, which impose heavy loads, performed by IdP server \( a \) are key exchange, key generation, signing, signature verification, encryption, and decryption. In these processes, IdP server \( a \) uses \( pk_y \) used in the HPET protocol, private key \( S_y^{(o)} \), and public keys in certificates \( I_s^{(o)}, I_s^{(p)}, I_y^{(p)}, I_x^{(p)}, \) and \( I_y^{(A)} \). The processing performance of an IdP server depends on key exchange and public key algorithms. For example, the processing performance of a server using TLS largely depends on key exchange and public key algorithms [19], [20], where public key algorithms are used for authentication. Frequently used major key exchange algorithms are Ephemeral Diffie-Hellman (DHE), Ephemeral Elliptic Curve Diffie-Hellman (ECDHE), and RSA. On the other hand, frequently used major authentication algorithms are RSA, DSA, and Elliptic Curve DSA (ECDSA). In the remainder of this paper, we represent the combination of a key exchange and an authentication algorithm by concatenating...
Table 3  Key length recommendations by NIST and IPA

<table>
<thead>
<tr>
<th>Key length for RSA</th>
<th>Curve for ECDSA</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>2048 bit (SHA256)</td>
<td>SECP224r1</td>
<td>NIST (~ 2030)</td>
</tr>
<tr>
<td>2048 bit (SHA256)</td>
<td>SECP256r1</td>
<td>IPA</td>
</tr>
<tr>
<td>3072 bit (SHA256)</td>
<td>SECP256r1</td>
<td>NIST (2031 ~)</td>
</tr>
</tbody>
</table>

their names with an underscore (_) in this order. For example, ECDHE_RSA denotes the combination of key exchange algorithm ECDHE and authentication algorithm RSA.

In major combinations of a key exchange and an authentication algorithm, the top-ranked combinations in terms of use frequency and TLS processing performance are ECDHE_RSA and ECDHE_ECDSA, respectively [19], [20]. When we use ECDHE_RSA and ECDHE_ECDSA, public key algorithms used in the PKI certificates are RSA and Elliptic Curve Cryptography (ECC), respectively. We select ECDHE_RSA for evaluating the HE-DOEV method because it is easy to deploy in current Internet environments. In addition, we assume that there is one intermediate CA between a root CA and an IdP, which affects the number of signature verification processes performed in the TLS.

In the procedures of the HE-DOEV method, we use the HPET protocol, which needs additive homomorphic cryptography. We assume that Paillier cryptography is used because it is one of the most common additive homomorphic cryptography systems.

In addition to key exchange and public key algorithms, the processing performance of an IdP server largely depends on the key length of public-key cryptography. NIST and the Information-Technology Promotion Agency (IPA) have recommended key lengths [21], [22]. For example, NIST recommends the combination of 2048-bit key for RSA and SHA-256 to be used through 2030 [21]. It also recommends SECP224r1 for ECDSA to be used through 2030 [21]. For use in 2031 and beyond, NIST recommends the combination of 3072-bit key for RSA and SHA-256 and SECP256r1 for ECDSA [21]. IPA recommends the combination of 2048-bit key for RSA and SHA-256 and SECP256r1 for ECDSA in 2015 [22]. Table 3 shows the above recommendations by NIST and IPA. To evaluate an IdP server, we use conditions 1 and 2 in Table 3, because we aim to measure the scale of the number of IdP servers when the proposed method is used in the near future. In addition to RSA and ECDSA, we use Paillier cryptography for the HPET protocol. For the key length of Paillier cryptography, we use the same key length as that for RSA.

4.2.2 Result and Discussion

Figure 8 shows the processing performance of an IdP server depending on the average data size of messages handled by an IdP server. We evaluated the processing performance of an IdP server using a Linux-based server whose specifications are shown in Table 4. The vertical and horizontal axes represent the processing performance of an IdP server and the data size of a message, respectively. The processing performance of an IdP server is measured and plotted for the PII data sizes of 16, 32, 64, 128, and 256 bytes in Fig. 8. As shown in the figure, the processing performance of an IdP server is almost equal in the range of data size from 1024 to 4096 bytes when the size of PII data is equal. In addition, the processing performances of RSA2048-SECP224r1 and RSA2048-SECP256r1, which correspond to 1 and 2 in Table 3, respectively, are almost the same, although that of the former is slightly better.

In Fig. 9, the solid lines show the processing performance of an IdP server depending on the size of PII data, which is changed from 16 to 256 bytes. The larger the PII data, the lower the processing performance of an IdP server.
The dashed lines in Fig. 9 show the processing performance of an IdP server when we use the IdP-to-IdP comparison for verifying the equality of PII data items associated with two devices, where the IdP-to-IdP comparison was discussed in Sect. 3.4. In this case, we assume that the HPET protocol is used between IdP servers. As described in Sect. 3.5, there are a chooser and a sender in the HPET protocol, and only a chooser can know the result of a HPET process. Therefore, each IdP server has to play both the roles of a chooser and a sender. As a result, the processing performance of the IdP-to-IdP comparison is much lower than that of the proposed method. The processing performance of the proposed method ranges from around $1.40 \times 10^2$ to $1.80 \times 10^2$ transactions per second (TPS).

In the proposed method, an IdP server issues a pseudonym certificate to a user device. At the issuance of a pseudonym certificate, an IdP server uses its private key. The private key should be managed in a strictly secure manner. To evaluate the previously proposed DOEV method, we measured its processing performance under the condition where pseudonym certificates were issued by using a hardware security module (HSM). The HSM we used was an nShield Connect 6000+ of Thales Inc.. The processing performance of signing to certificates when using this HSM was around 1.50×10⁷ TPS [10]. This processing performance is roughly close to that of the proposed method when the size of PII data is 256 bytes. We here consider a unit that consists of an IdP server and an HSM. We call this unit a 1-to-1 unit. In the following discussion, we assume that the processing performance of the proposed method provided by a 1-to-1 unit is $1.40 \times 10^2$ TPS, which corresponds to the processing performance for 256 byte sized PII data.

We next discuss the user capacity of a 1-to-1 unit on the basis of its processing performance when the size of PII data is 256 bytes. The user capacity of a 1-to-1 unit means the number of users whose HE-DOEV requests can be processed by a single 1-to-1 unit. Let $U$, $C$, and $P$ be the user capacity of a 1-to-1 unit, the number of times that a user requests HE-DOEV processes in a year, and the processing performance of a 1-to-1 unit, respectively. In addition, let $Y$ be the time of one year in the unit of a second. Then, the number of HE-DOEV processes requested to a single 1-to-1 unit in one second is $U \cdot C / Y$ when we assume that users request HE-DOEV processes equally at any time. On the other hand, the usage rate of a computer and the Internet depends on time. According to the Ministry of Internal Affairs and Communications, the peak and the average usage rate of the Internet in Japan are 28% and 15%, respectively [23]. When we assume that the rate of the peak usage rate against the average one $r = (28/15)$ is equal to that in the usage rate of an HE-DOEV service, the peak usage rate of an HE-DOEV service is $U \cdot C \cdot r / Y$. This peak is equal to $P$, which is the processing performance of a 1-to-1 unit. Therefore, we have

$$U = \frac{P \cdot Y}{C \cdot r}$$

We can calculate the user capacity of a 1-to-1 unit $U$ using the above equation.

Figure 10 shows the user capacity of a 1-to-1 unit, which is depicted by solid lines. The vertical and horizontal axes in the figure are the user capacity of a 1-to-1 unit ($U$) and the number of times that a user requests HE-DOEV processes in a year ($C$), respectively. We also plotted the user capacity of a 1-to-4 unit that consists of an IdP server using the previously proposed DOEV method and four HSMs. This is depicted by dashed lines in the figure. In our previous paper, we combined one IdP server and four HSMs as a unit because the processing performance of an IdP server using the previously proposed method is around four times that of signing to certificates achieved by the same HSM as used in the evaluation of that paper [10]. From Fig. 10, we find that the processing performance of a 1-to-1 unit using the HE-DOEV method is around ¼ that of a 1-to-4 unit using the previously proposed DOEV method. This is because the role of a sender in the HPET protocol imposes a heavy load on an IdP server. We next discuss the scale of a system to provide a DOEV service using the HE-DOEV method to 100 million users. When a user requests HE-DOEV processes 50 and 100 times in a year, the processing performances of a 1-to-1 unit are $4.60 \times 10^7$ and $2.30 \times 10^7$, respectively. Therefore, the numbers of 1-to-1 units needed to provide a service to 100 million users are 3 and 5, respectively. This means that a DOEV service can be provided with a small-scale system even when using the HE-DOEV method.

As described in Sect. 4, the HE-DOEV method proposed in this paper is more advantageous than the previously proposed DOEV method in terms of privacy, because the HE-DOEV method eliminates the disclosure of PII data items to other IdPs and user devices. On the other hand, the HE-DOEV method has lower processing performance than the previously proposed DOEV method. Therefore, we can select either of these methods in accordance with whether privacy or cost is a higher priority for a service.

Finally, we mention how much user experience is affected by processes in the HE-DOEV method. In the HE-
Table 5  Execution times of RSA-related functions with and without TEE

<table>
<thead>
<tr>
<th>Functions (RSA)</th>
<th>Without TEE (s)</th>
<th>With TEE (s)</th>
<th>Rate (T₂/T₁)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Key generation</td>
<td>3.18 × 10⁻¹</td>
<td>1.08</td>
<td>3.40</td>
</tr>
<tr>
<td>2) Encryption</td>
<td>1.61 × 10⁻⁴</td>
<td>3.76 × 10⁻⁴</td>
<td>2.33</td>
</tr>
<tr>
<td>3) Decryption</td>
<td>3.58 × 10⁻³</td>
<td>2.26 × 10⁻²</td>
<td>6.31</td>
</tr>
</tbody>
</table>

Fig. 11  Times needed for key generation, encryption, and decryption processes using Paillier cryptography

DOEV method, a user device needs to 1) generate a pair of a public key and private key, 2) encrypt PII data for a UVR message, and 3) decrypt data in a received PCR message in its TEE. Since such processes regarding public-key cryptography generally impose a heavy load on a computer, we roughly evaluated the proposed method to estimate the effect on user experience. In this evaluation, we estimated times needed for processes 1), 2), and 3) through two steps: a) measuring the difference in processing performances between applications running on a normal OS and in a TEE, and b) estimating times needed for processes 1), 2), and 3) that run in a TEE. In step a), we measured times needed for the processes of key generation, encryption, and decryption provided by RSA-related functions. In this measurement, we used a commercially available smartphone (Samsung S8 (SM-G950F)), which provided RSA-related functions with and without a TEE. Table 5 shows the result of step a). The processes of RSA-related functions with a TEE took 2.33 to 6.31 times longer than those without a TEE. In step b), we first measured times needed for processes 1), 2), and 3) by using the functions of Paillier cryptography without a TEE, where we used the same commercially available smartphone as used in the evaluation of step a). Figure 11 shows the measured times needed for these processes, where the key length of Paillier cryptography is 2048 bits, and we assumed that the size of PII data is 256 bytes. As shown in this figure, the times needed for processes 1), 2), and 3) are all less than 1 s. However, process 1) (key generation) needs a comparatively longer time, which is 3.48 × 10⁻¹ s. We here assumed that processes 1), 2), and 3) can take 3.5 and 6.5 times longer than the measured times if they are executed in a TEE. Figure 11 also shows these estimated times.

An encryption and a decryption process take less than 1 s even if we assume that they can take 6.5 times longer than the measured times in step a), whereas process 1) (key generation) takes more than 2 s. In the HE-DOEV method, we can prepare one or more key pairs of a public key and private key in advance before a user invokes an HE-DOEV process because we can execute process 1) at any time for the future use of key pairs. In addition, methods have been proposed to improve encryption performance (throughput) of Paillier cryptography, one of which exhibits encryption performance more than 100 times higher than that of naïve implementation [24]. Therefore, we estimate that the HE-DOEV method can offer a satisfactory user experience.

5. Related Work

In this paper, we proposed the HE-DOEV method to verify the equality of device owners using homomorphic encryption. This method is used for credential sharing among user devices. There are several existing methods for credential sharing among user devices. Examples include password management tools provided by LogMeIn, Inc. [25], McAfee, LLC [26], and TrendMicro, Inc. [27]. Another method for credential sharing is the iCloud key chain provided by Apple, Inc. [28]. Most of such password management tools encrypt credentials using a master password and store them as ciphertext. Therefore, a user can retrieve a password by entering a master password he/she has memorized. In the iCloud key chain, when a user obtains a new device in addition to a current one and shares credentials stored in a current device to a new one, a new device is required to log in to the iCloud service using the user’s Apple ID, to input a number called a security code, and to obtain admission for adding a new device from a current one. In the iCloud key chain, credentials in one device are transmitted to another in an encrypted form. These existing methods are designed on the basis of WYK authentication methods.

In the secure-copy based credential sharing method, we assume that a DOC is issued and installed in the SE/TEE of a user device when a user obtains it. Prior to the issuance of a DOC, a device provider verifies who the owner of a device is by using his/her ID cards. By binding a user device to its owner, we make it difficult for attackers to steal credentials in user devices. The secure-copy based credential sharing method makes it even more difficult for attackers to steal credentials by forcing a new device to obtain the admission of a current one to share its credentials, similarly to the iCloud key chain. In addition, we can disable a credential sharing service to transmit credentials from a lost device to another device when we use the HE-DOEV method, because IdP cooperates with user devices in the proposed method.

NIST defines a method to issue secure and reliable credentials to users’ smartphones, which is designed for US federal government employees possessing personal identity verification (PIV) cards [29]. In this definition, a PKI certificate is issued to a smartphone through an identity proofing process using the PIV card of its owner. The Japanese gov-
ernment issues individual number cards called My Number cards [30]. In addition, investigations are currently being conducted into how to issue a new credential to a smartphone and improve the usability of My Number cards, although the number of user devices to which new credentials are issued is limited to one. These methods do not satisfy requirements in multi-device environments because it is forbidden to issue credentials to several devices.

In addition to the password management tool and the iCloud key chain described above, a method has been proposed to issue a new credential to use a service on the basis of an already registered credential, which is called a transfer access protocol [31]. In this method, a credential that is newly generated in a new device is registered to an authentication server. In this registration process, a new key is signed with a private key corresponding to an already registered public key. This method does not define how to verify the equality of owners of devices. The HE-DOEV method proposed in this paper can be used in cooperation with the transfer access protocol.

6. Conclusion

In this paper, we proposed a new homomorphic encryption based device owner equality verification (HE-DOEV) method to verify whether the owners of two devices are equal. In the HE-DOEV method, we assume that the owner of every device is associated with a public key infrastructure (PKI) certificate issued by an identity provider (IdP), where a PKI certificate is used to authenticate the owner of a device. The proposed method verifies device owner equality under the condition where multiple IdPs can issue PKI certificates to user devices. In addition, it verifies device owner equality without disclosing any privacy-related information managed by an entity to others by using homomorphic encryption. We evaluated the processing performance of a server needed for an IdP in the HE-DOEV method. The evaluation showed that the HE-DOEV method can provide a device owner equality verification service for 100 million users by using a small-scale system in terms of the number of IdP servers.

References

[29] H. Ferraiolo, D. Cooper, S. Francomacaro, and A. Regenscheid, “Guidelines for Derived Personal Identity Verification (PIV) Cre-


Yoshihiko Omori received his M.E. degree in electrical communication engineering from Tohoku University in 1993. Since joining NTT in 1993, he has been engaged in research on traffic control in IP-based networks, QoS control, operation systems for VPNs, authentication technologies, and packet networks for mobile communications at NTT Telecommunication Networks Laboratories and NTT DOCOMO. He is a member of IEICE.

Takao Yamashita received his B.S. and M.S. degrees in electrical engineering in 1990 and 1992 from Kyoto University, where he also received his Ph.D. degree in informatics in 2006. In 1992, he joined NTT. His current research interests encompass network security, cloud computing, Internet-of-Things, and distributed algorithms. He is a member of IEICE, IEEE, IPSJ, and APS.