PAPER Special Section on Enriched Multimedia—Creation of a New Society through Value-added Multimedia Content—

Application Authentication System with Efficiently Updatable Signature*

Kazuto OGAWA[†a)] ***and*** **Go OHTAKE**[†,††]**,** ***Members***

SUMMARY Broadcasting and communications networks can be used together to offer hybrid broadcasting services that incorporate a variety of personalized information from communications networks in TV programs. To enable these services, many different applications have to be run on a user terminal, and it is necessary to establish an environment where any service provider can create applications and distribute them to users. The danger is that malicious service providers might distribute applications which may cause user terminals to take undesirable actions. To prevent such applications from being distributed, we propose an application authentication protocol for hybrid broadcasting and communications services. Concretely, we modify a key-insulated signature scheme and apply it to this protocol. In the protocol, a broadcaster distributes a distinct signing key to each service provider that the broadcaster trusts. As a result, users can verify that an application is reliable. If a signed application causes an undesirable action, a broadcaster can revoke the privileges and permissions of the service provider. In addition, the broadcaster can update the signing key. That is, our protocol is secure against leakage of the signing key by the broadcaster and service providers. Moreover, a user terminal uses only one verification key for verifying a signature, so the memory needed for storing the verification key in the user terminal is very small. With our protocol, users can securely receive hybrid services from broadcasting and communications networks.

key words: *hybrid services through broadcasting and communications networks, application authentication, ID-based signature, key-insulated signature*

1. Introduction

1.1 Background

Several hybrid services combining the functionalities and resources of broadcasting and communications networks have been developed. Hulu [15] in the US is an online video service that offers TV programs and movies through the Internet. All content is provided by broadcasters, such as NBC, FOX, and ABC, and movie companies. HbbTV [13], [14] is a pan-European initiative aimed at harmonizing broadcast and broadband delivery of entertainment through digital TVs and set-top boxes. Its services include video on demand (VoD) as well as program-related services such as digital text and electronic program guides (EPGs). The founding members of the HbbTV consortium consist of European television broadcasters and consumer electronics companies. YouView [22] in the UK is a hybrid service that offers high-definition TV, catch-up TV, and Internet services such as YouTube and Facebook through digital TVs and set-top boxes. YouView is jointly being developed by broadcasters (BBC, ITV, Channel 4, and Channel 5) and information and communication companies (TalkTalk, BT, and Arqiva). Korean broadcasters (KBS, MBC, SBS, and EBS) started Open Hybrid TV (OHTV) in 2013 that combines terrestrial digital TV and Internet [8], [19]. They are cooperating with TV manufacturers, such as Samsung Electronics, LG Electronics, and Net&TV, and academia in standardization of OHTV. In OHTV, broadcasters provide users with services such as advanced EPGs, VoD, video bookmarking, advertising, etc. In Japan, NHK and commercial broadcasters have launched Hybridcast [1], [16]–[18], which leverages the functions of communications networks to enhance existing digital broadcasting services to provide customization, social networking, related program recommendations, and interaction with portable devices. In addition, other hybrid services, such as Smart TV Box for cable TV [20] and SyncCAST [21], are being developed.

Hybrid services have to run many applications on a user terminal. Moreover, to offer attractive services to users, it is necessary to establish an environment in which any service provider can create applications and distribute them to the users. However, malicious service providers might distribute applications that cause the user terminal to take undesirable actions. To thwart such malicious service providers and assure users that they can use an application securely, *application authentication* can be used to verify that an application is originated from a trusted service provider and has not been modified in any way. The hybrid services described above currently do not use any authentication scheme.

Digital signature schemes are used for a lot of authentication purposes. In application authentication, a service provider signs an application with its signing key and distributes the application to user terminals. Each user terminal receives the application and verifies the signature by using the corresponding verification key. The trouble is that if the signing key is leaked to an adversary, he or she can easily impersonate the service provider. To prevent this from happening, it must be easy to update the signing key. If malicious service providers give user terminals signed applications that cause undesirable actions to occur, the service

Manuscript received March 26, 2015.
Manuscript revised July 22, 2015.
Manuscript publicized October 21, 2015.

[†]The authors are with Japan Broadcasting Corporation, Tokyo, 157–8510 Japan.
[††]The author is with University of Calgary, Canada.

*Preliminary version of this paper was presented at The 12th International Workshop on Information Security Applications (WISA 2011) [10].

a) E-mail: ogawa.k-cm@nhk.or.jp

DOI: 10.1587/transinf.2015MUP0008

provider's privileges and permissions must be revoked by using the Certificate Revocation List (*CRL*). Since hybrid services will likely have a large number of service providers, a countermeasure must take into account the ease of signing key updates, the need for efficient transmission of the huge *CRL*, and the huge memory taken up by storing the verification key in a user terminal.

1.2 Our Contribution

We developed an application authentication protocol for hybrid services that take advantage of broadcasting and communications networks. We modified a key-insulated signature (KIS) scheme [9] and applied it to the protocol. In particular, a temporal identity is issued to a service provider that is composed from the time period over which the single signing key is valid and the provider's real identity, and it is used as a time period in a KIS scheme. This modification enables KIS schemes to be applied to the application authentication protocol for hybrid services.

If a signed application causes some undesirable action, a broadcaster can easily revoke the privileges and permissions of the service provider by putting its temporal identity in the *CRL* and its identity in an update prohibited list (UPL). When a broadcaster updates the signing key of a service provider, it transmits a partial key used for such updates to the service provider. This enables broadcasters to update signing keys online. Moreover, the integrity of two dimensions, i.e., the time period and provider identity, enables us to simplify the structure of the *CRL*. That is, we make it so that the size of the *CRL* is not in proportion with the number of revoked providers.

A broadcaster distributes the provider's signing keys to providers whom it trusts. Through the protocol, users can verify that an application is reliable. A user terminal uses only one verification key for verifying signatures, so the memory needed for storing the verification key in a user terminal is very small. In addition, the verification key is rarely updated. This makes the communication cost of verification key update very small.

In hybrid services, a broadcaster can use broadcast channels, such as the airwaves, as secure channels for distributing large amounts of data to all users simultaneously. Our protocol is more suitable for hybrid services than for Internet services since a broadcaster can distribute a *CRL* to users through secure and efficient broadcast channels. In addition, the broadcast channel can be used when the verification key is updated. Though such updates occur infrequently, the key can be broadcast to all user terminals efficiently, simultaneously, and securely.

By using our application authentication protocol, users can securely receive hybrid broadcasting and communications services.

1.3 Organization

The rest of the paper is organized as follows. Section 2 shows several models for hybrid services combining the broadcasting and communications networks. Section 3 introduces the requirements for the services. In Sect. 4, we reviews ID-based signature (IBS) schemes and KIS schemes, and shows generic construction of application authentication protocols from these signature schemes. In Sect. 5, we focus on the security and efficiency of the protocol, and constructs a practical application authentication protocol from a specific KIS scheme. We finally evaluate its performance and show that the proposed protocol is more secure and efficient than the other protocols.

2. Model

Generally, there are two models for authentication using a digital signature scheme: one model is where a signer generates a pair of signing and verification keys and distributes the verification key to a verifier (Model 1); the other model is where a certain authority (manager) generates the key pair and distributes the signing key to a signer and the verification key to a verifier (Model 2). Figures 1 and 2 show these models for hybrid services. In each model, a service provider transmits an application to a user terminal through the Internet in answer to a user request. The service provider adds its signature to the applications and transmits them to the user. The user can verify the signature with the corresponding verification key in the user terminal and make sure that the application originated from the service provider and has not been modified in any way.

Figure 1 shows Model 1, where the service provider generates signing and verification keys. This model can, for example, use the Public Key Infrastructure (PKI). The service provider generates a pair of keys and gets a certificate corresponding to the verification key from a broadcaster/Certificate Authority (CA), which is a trusted third party and is a broadcaster in this case. Service providers A and B get certificates corresponding to their verification keys from the broadcaster. Service provider C does not have a certificate. A and B can add their signature to applications with their signing key, so the user terminal can verify the signatures by using their verification keys with the certificates and make sure that the service provider certified by the broadcaster has made a signed application.

Figure 2 shows Model 2, where the broadcaster generates a pair of signing and verification keys. The broadcaster distributes the signing key to only a trusted service provider in advance. The difference from Fig. 1 is that only one entity generates the key pair. The broadcaster distributes signing keys to A and B, which are trusted, but it does not distribute one to C, which is not trusted. The subsequent processes of A, B, and C are the same as those of Fig. 1. The service providers certified by the broadcaster make signed applications.

3. Requirements

To perform application authentication securely and ef-

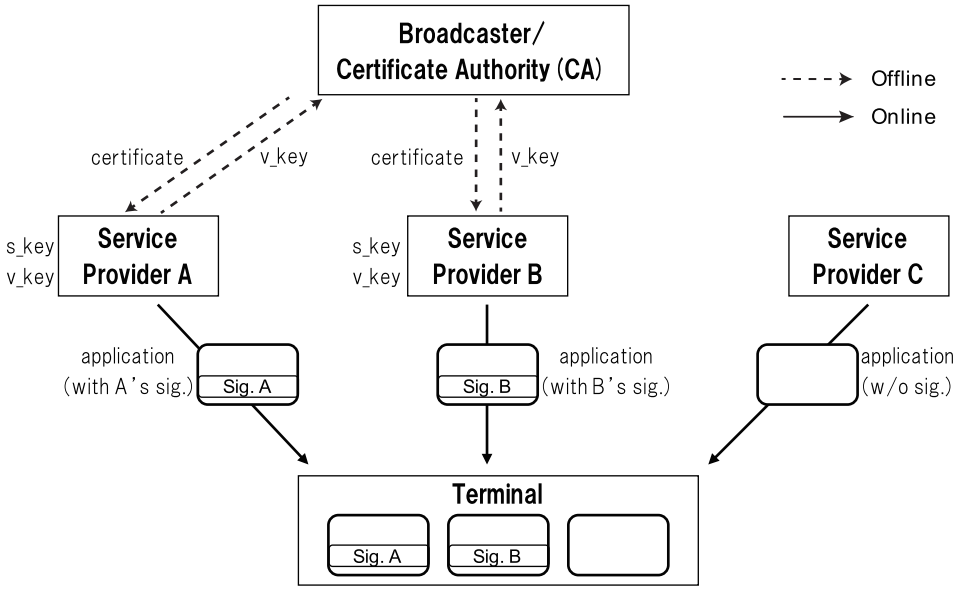

Fig. 1 Application authentication model in hybrid services of broadcasting and communications networks in which each service provider generates keys. V_key and s_key denote the verification key and signing key, respectively. Sig.A and Sig.B denote A and B's signatures.

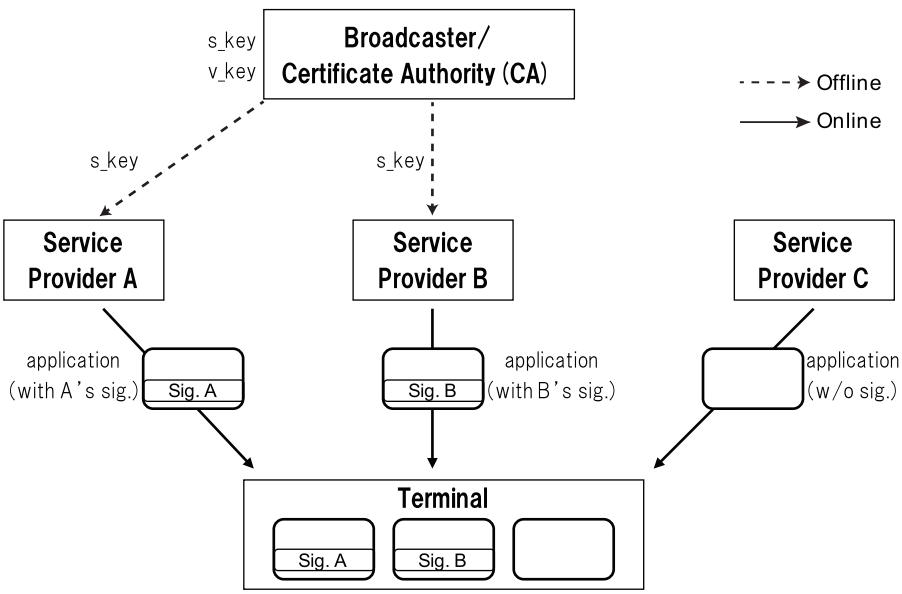

Fig. 2 Application authentication model of hybrid services offered through broadcasting and communications networks in which a broadcaster generates keys.

ficiently in hybrid services, the following requirements should be satisfied.

(1) *The number of stored verification keys must be small.*
Verification keys must be stored in a user terminal. If the number of keys were to grow in the future, the memory needed for storing them might become huge. Therefore, the number should be kept as small as possible.

(2) *Signatures must be verified efficiently.*
Each user terminal has at least one CPU, and the CPU has to execute a lot of procedures for the terminal to provide services to users. The CPU cannot prioritize an application authentication procedure over any other procedures directly involved in the service. This means the CPU cost of the verification process should be as small as possible.
In addition, it is preferable that there is no communication with other entities for the verification, because such a communication causes delays that might degrade the quality of service.

(3) *Service providers must be revoked efficiently.*
If a malicious service provider posts an application that causes a user terminal to take an undesirable action, the broadcaster must revoke the service provider's privileges and do so efficiently.

(4) *The signing key must be easily updated.*
Although a service provider must manage its signing key securely, the signing key might nonetheless be leaked to an adversary as the result of a server attack. Therefore, the signing key must be easy to update.

Let us explain the second requirement in detail here. When a certain application is downloaded, authentication is necessary for the sake of security. Consider a VoD service application that should be resident in a user terminal in order to be usable. This means that it is authenticated only once when it is downloaded or upgraded. Such upgrades tend to occur on a monthly basis for TV sets and STBs, similar to PC, mobile phone and mobile terminal applications. An update once a month does not seem to be heavy cost, but there are applications that require quick and low cost authentication. This tends to be the case for applications that make TV programs more attractive, especially sport programs, and there could be many of them operating during the program. For instance, one application may introduce the players, while another may analyze the plays of each player. Moreover, such applications are developed for individual programs. They need to be downloaded at the beginning of the program or before the program, and are removed after the end of the program. The number of applications for a program is not restricted, and dozens of applications are expected. Even if the number of applications used by a user is not so large, sometimes such applications would be downloaded in the middle of the program. In such a situation, quick downloading, quick authentication, and quick start are required, because users do not want to wait a long time. Hence, low cost and quickness are important factors for authentication.

Let us revisit Model 1 and 2 in the context of the above requirements.

In Model 1 shown in Fig. 1, the broadcaster certifies the relation between a service provider and its verification key, which means that a verification key is assigned to each service provider. This idea is the same as that of the current PKI. Model 1 has three variants.

Model 1-1. Each terminal stores all service providers' verification keys. The keys are huge and require an enormous amount of storage. This variant does not satisfy requirement (1).

Model 1-2. The terminal does not store any verification key. It has to verify signatures by communicating through networks whenever it receives an application with signatures. In addition, at least two verifications are necessary: one for the signature on the application, the other for the signature on the certificate. It should be noted that, if the certificate issued by a trusted third party is not attached, the signature is meaningless. The CPU cost of this verification, which includes calculation and communication costs, is substantial; hence, this variant does not satisfy requirement (2).

Model 1-3. The terminal stores only the broadcasters' verification keys. The users cannot determine the broadcasters that the service providers use and it is assumed that multiple broadcasters are used. However, the number of broadcasters is not so large and the number of stored keys is small. This variant satisfies requirement (1). It requires at least two signature verifications to authenticate an application: one for the signature on the application, the other for the signature on the certificate. This seems to be possible, and thus, it would seem to satisfy requirement (2). It should be noted that, when the number of applications that run simultaneously in a terminal is m, $2 \times m$ verifications are necessary.

In Model 2 shown in Fig. 2, the broadcaster distributes signing keys to the service providers that it trusts. Even if the broadcaster trusts the providers, it is not a secure practice to distribute identical signing keys to multiple providers, and hence, a distinct signing key should be distributed to each provider. In cases that a general signature scheme that is used in the current PKI (e.g. DSA, ECDSA, RSA-PSS, and RSASSA signature schemes) is used, there are three variations that are the same as those of Model 1.

Model 2-1. Each terminal stores all service providers' verification keys.

Model 2-2. The terminal does not store any verification key.

Model 2-3. The terminal stores only the broadcasters' verification keys.

Although these characteristics are the same as those of Model 1-1, Model 1-2, and Model 1-3, the signature schemes [2]–[4], [6], [7], [9], [11], [12] that have only one verification key for many signing keys can be used in this model. Hence, requirement (1) can be satisfied by using such schemes. The terminal only has to make one verification per application and, in this sense, the model satisfies requirement (2). It should be noted that, when the number of applications that run simultaneously is m, m verifications are necessary.

Basically, at least 2 and 1 verifications are necessary in Model 1 and Model 2, respectively. It should be noted that the system of Model 1-1 requires only one verification, but the number of verification keys stored in each terminal is large and the model is not practical. When the number of applications is m, $2 \times m$ and m verifications are required for each model. It is impossible to determine the actual CPU costs of the terminal beforehand since it is impossible to correctly estimate the number m. Hence, we cannot say that the number of verifications should be 1 or that $2 \times m$ verifications are acceptable. Only we can say that it is preferable that the number of verifications is smaller. The above observations indicate that we should choose only the model shown in Fig. 2.

4. Generic Construction of Application Authentication from ID-Based Signature and Key-Insulated Signature Schemes

In Model 2 shown in Fig. 2, the broadcaster distributes a different signing key to each service provider. Moreover, it has to distribute only one verification key to users, as per requirement (1) in Sect. 3. IBS schemes [2], [3], [6], [11], [12] and KIS schemes [4], [7], [9] can satisfy this requirement, as they have one verification key for many signing keys. In this section, we review these signature schemes, describe two generic constructions of the application authentication protocol that apply them to Model 2, and evaluate these constructions from viewpoints of the other requirements.

4.1 ID-Based Signature Scheme

IBS schemes are developed in order to verify a signature by using a signer's identity, such as name and e-mail address. In this scheme, a trusted authority issues a signing key corresponding to a user identity and only one verification key for the system. The verification key can be a common system parameter. The verifier uses the verification key and the signer's identity for verifying a signature. The verification key is unchanged and the signer's identity is included in the signature.

An IBS scheme [2], [3], [6], [11], [12] consists of four polynomial-time algorithms (Setup, Extract, Sign, Vrfy).

- Setup: This is a probabilistic algorithm that takes as input a security parameter 1^λ. It returns a verification key vk and a master key msk.
- Extract: This is a probabilistic algorithm that takes as inputs msk, vk, and a user identity id_i ($i \in \{1, \cdots, m\}$). It returns the corresponding signing key sk_{id_i}.
- Sign: This is a probabilistic algorithm that takes as inputs sk_{id_i} and a message M. It returns a signature σ_i.
- Vrfy: This is a deterministic algorithm that takes as inputs vk, id_i, M, and σ_i. It returns a bit b, where $b = 1$ means that a signature is accepted.

4.2 Key-Insulated Signature Scheme

KIS schemes are developed in order to minimize the damage resulting from signing key exposure by employing signing key update and giving signing keys a temporal property. They do not need to update their verification key even if their singing keys have to be updated. That is, their verification keys are unchanged.

A KIS scheme [4], [7], [9] consists of five polynomial-time algorithms (Gen, Upd*, Upd, Sign, Vrfy).

- Gen: This is a probabilistic algorithm that takes as inputs a security parameter 1^λ and the total number of time periods T. It returns a verification key vk, a master key msk, and an initial signing key sk_0.
- Upd*: This is a probabilistic algorithm that takes as inputs msk and indices t_i and t_j for time periods (we assume that $1 \le t_i, t_j \le T$). It returns a partial key sk'_{t_i,t_j}.
- Upd: This is a deterministic algorithm that takes as inputs indices t_i and t_j, a signing key sk_{t_i} for time period t_i and sk'_{t_i,t_j}. It returns a signing key sk_{t_j} for time period t_j.
- Sign: This is a probabilistic algorithm that takes as inputs an index t_j, a message M, and sk_{t_j}. It returns a pair $\langle t_j, \sigma_j \rangle$ consisting of an index t_j for a time period and a signature σ_j.
- Vrfy: This is a deterministic algorithm that takes as inputs M, $\langle t_j, \sigma_j \rangle$, and vk. It returns a bit b, where $b = 1$ means that the signature is accepted.

In this scheme, a signer updates his or her signing key by using a time period. Therefore, a signing key has an expiration time, and the damage caused by signing key leakage can be limited to being within that particular time period.

4.3 Generic Construction from ID-Based Signature Scheme

Figure 3 shows a generic construction of an application authentication protocol using the IBS scheme in Model 2. The user identity id_i ($i \in \{1, \cdots, m\}$) is substituted with a provider identity p_i ($i \in \{1, \cdots, m\}$). A broadcaster runs Setup and generates a verification key vk and a master key msk. The verification key vk is published. The broadcaster then runs Extract(msk, vk, p_i) and distributes a signing key sk_{p_i} for a trusted service provider p_i ($i = 1, 2, \ldots, m$) offline. The service provider p_i runs Sign(sk_{p_i}, M) for an application M and transmits the signed application (M, σ_i) to user terminals. The user terminals receive it and run Vrfy(vk, p_i, M, σ_i).

In Fig. 3, a user terminal needs only one verification key vk for application authentication. Therefore, requirement (1) is satisfied. The number of verification is only one, and therefore, requirement (2) is satisfied. If the privileges of a service provider have to be revoked, a broadcaster creates a *CRL* including the identities of the service providers whose privileges are to be revoked and then distributes it to all user terminals. A user terminal checks the *CRL* before it verifies the received signature. If the signature is created by a service provider whose identity is in the *CRL*, the terminal can determine that the signature is invalid. Thus, requirement (3) is satisfied. However, when updating a signing key, a broadcaster must assign a new identity to each service provider, run Extract to generate the corresponding signing key, and distribute it to each service provider offline. This process incurs a heavy load. Therefore, requirement (4) is not necessarily satisfied.

4.4 Generic Construction from Key-Insulated Signature Scheme

Figures 4 and 5 show a generic construction of an application authentication protocol using the KIS scheme in Model 2. The time period t_i is substituted with a temporal identity $p_{i,u}$ of a service provider p_i, where p_i ($i \in \{1, \cdots, m\}$,

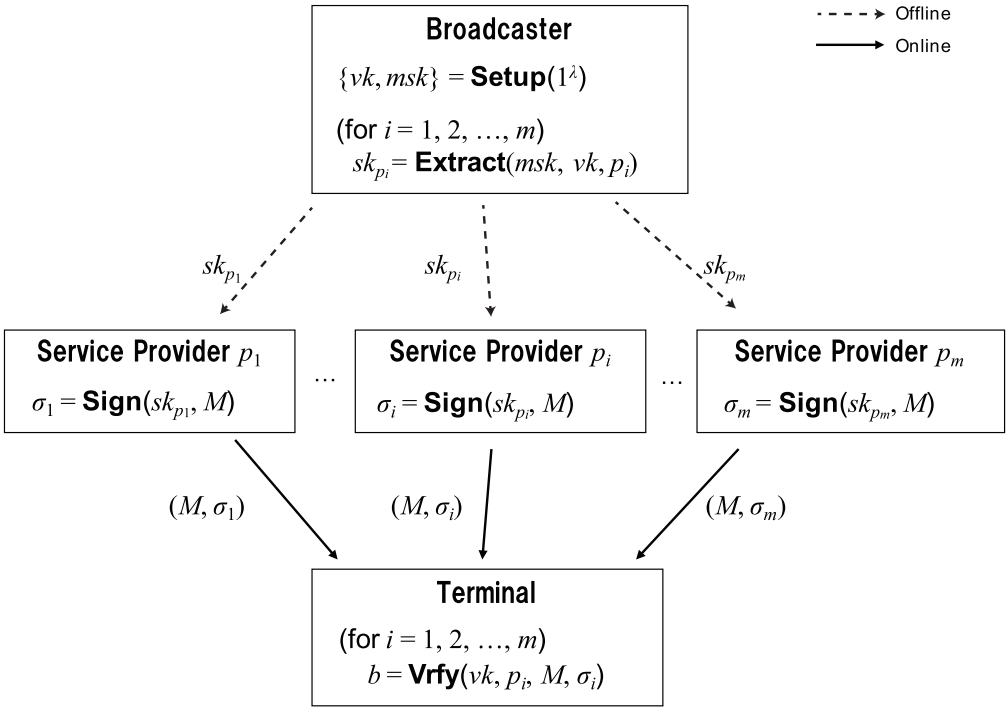

Fig. 3 Application authentication protocol using IBS

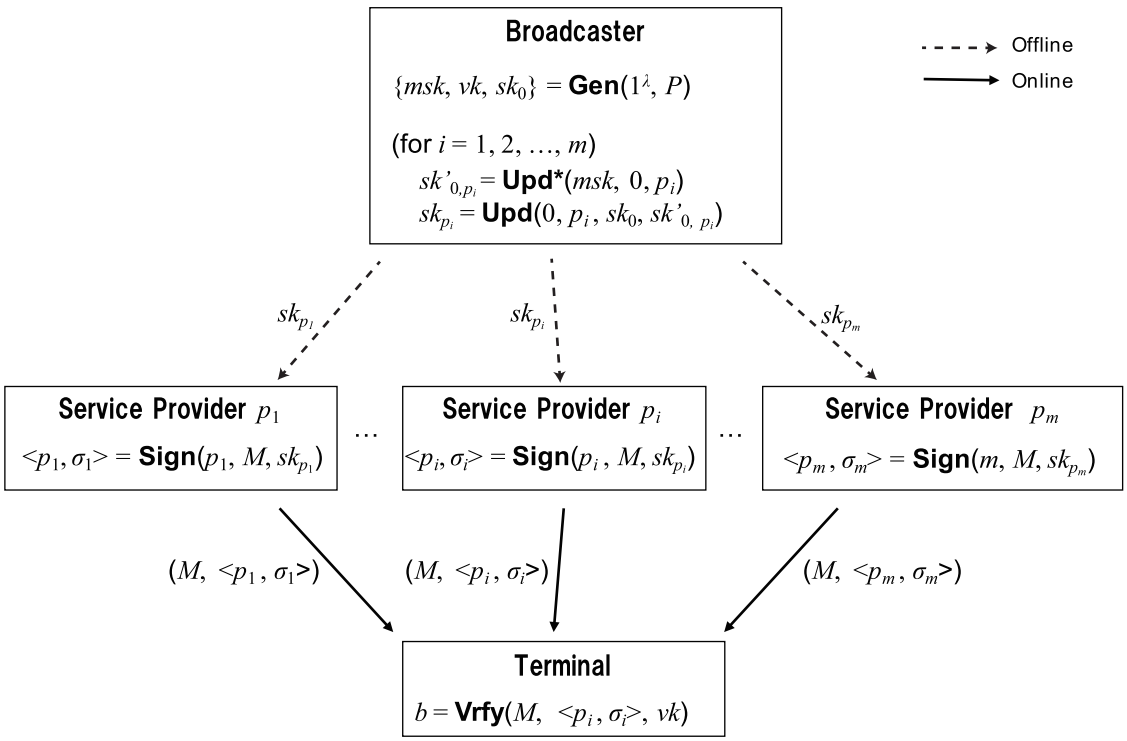

Fig. 4 Application authentication protocol using KIS: signing key generation and signing

$m \leq n$) denotes a provider's ID, m is the number of service providers, n is the maximum number of service providers and the index $u \in \{1, 2, \cdots\}$ denotes the number of signing key update. T is substituted with P that denotes the maximum number of $p_{i,u}$. Without loss of generality we set $p_{i,0} = p_i$ for the simplicity of the following explanation.

A broadcaster runs Gen and generates a master key msk, a verification key vk, and an initial signing key sk_0. The verification key vk is published. After that, the broadcaster runs $\mathsf{Upd}^*(msk, 0, p_i)$ to create a partial key sk'_{0,p_i}

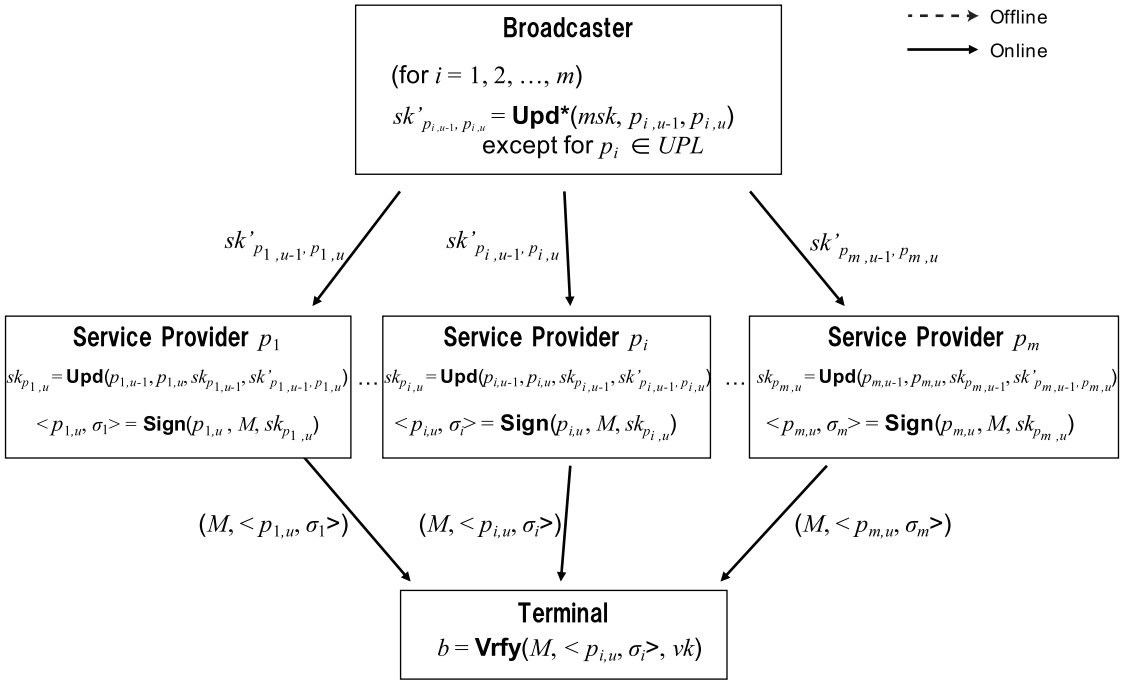

Fig. 5 Application authentication protocol using KIS: signing key update

and $\mathsf{Upd}(0, p_i, sk_0, sk'_{0,p_i})$ to create a signing key sk_{p_i}, and it distributes the signing key to a trusted service provider p_i offline. To add a trusted service provider p_i ($m < p_i \le n$), the broadcaster runs $sk'_{0,p_i} = \mathsf{Upd}^*(msk, 0, p_i)$ and $sk_{p_i} = \mathsf{Upd}(0, p_i, sk_0, sk'_{0,p_i})$ to create a signing key sk_{p_i} and sends the signing key to the service provider offline. When a service provider p_i distributes an application M to users, it runs $\mathsf{Sign}(p_i, M, sk_{p_i})$ to create a signature pair $\langle p_i, \sigma_i \rangle$ and transmits the signed application $(M, \langle p_i, \sigma_i \rangle)$ to the user terminals. The user terminals receive it and run $\mathsf{Vrfy}(M, \langle p_i, \sigma_i \rangle, vk)$.

To update a service provider's signing key, the broadcaster generates a temporal identity for the service provider by using two-dimensional elements consisting of the number of signing key update $u = 1, 2, \ldots$ and the real identity of a service provider p_i. That is, the temporal identity $p_{i,u}$ of the service provider p_i is

$$p_{i,u} = u \times n + p_i \ .$$

Figure 5 shows the procedure of updating the signing key $sk_{p_{i,u-1}}$, that denotes the signing key of a service provider p_i after the $(u-1)$th key update, to a new signing key $sk_{p_{i,u}}$. Signing key updates are performed for all service providers except for the providers that are included in a UPL, where the UPL is a list of service provider identities p_i ($i \in \{1, \cdots, m\}$) whose signing keys should not be updated. Only the broadcaster has the UPL and it is not transmitted to user terminals. The broadcaster runs $\mathsf{Upd}^*(msk, p_{i,u-1}, p_{i,u})$ to create a partial key $sk'_{p_{i,u-1}, p_{i,u}}$ and sends it to a service provider p_i. The service provider p_i then runs $\mathsf{Upd}(p_{i,u-1}, p_{i,u}, sk_{p_{i,u-1}}, sk'_{p_{i,u-1}, p_{i,u}})$ to create a new signing key $sk_{p_{i,u}}$. It should be noted that the service providers that are included in the UPL cannot get their partial keys and cannot update their signing keys. Therefore, these providers will not be able to generate any valid signature after this signing key update. When these providers would like to restart their services, they have to obtain new signing keys or partial keys offline (or through secure channels). When distributing an application M to a user, the service provider p_i runs $\mathsf{Sign}(p_{i,u}, M, sk_{p_{i,u}})$ to create a signature pair $\langle p_{i,u}, \sigma_i \rangle$ and transmits the signed application $(M, \langle p_{i,u}, \sigma_i \rangle)$ to the user terminal. The user terminal receives it and runs $\mathsf{Vrfy}(M, \langle p_{i,u}, \sigma_i \rangle, vk)$.

In Figs. 4 and 5, the user terminal needs only one verification key vk to make an application authentication. Therefore, requirement (1) is satisfied. The number of verification is only one, and therefore, requirement (2) is satisfied. If the privileges of some service providers have to be revoked, the broadcaster creates a CRL including their temporal identities and creates a UPL including their identities. The broadcaster then distributes the CRL to all user terminals. It should be noted that the UPL is not distributed to the terminals. For example, to update signing keys $sk_{p_{1,u-1}}, sk_{p_{2,u-1}}, \cdots, sk_{p_{m,u-1}}$ to $sk_{p_{1,u}}, sk_{p_{2,u}}, \cdots, sk_{p_{m,u}}$ as shown in Fig. 5, the broadcaster sets

$$CRL = \{u \times n\}$$

so that the old signing keys $sk_{p_{1,0}}, sk_{p_{2,0}}, \cdots, sk_{p_{m,0}}, \cdots, sk_{p_{1,u-1}}, \cdots, sk_{p_{m,u-1}}$ cannot be used[†]. Here, the element, $u \times n$, of the CRL means that the privileges of every service provider whose temporal identity is $p_{i,u} \in \{1 \le p_{i,u} \le u \times n\}$

[†]A broadcaster may set $CRL = \{(u-1) \times n + m\}$ so that the old signing keys $sk_{p_{1,0}}, sk_{p_{2,0}}, \cdots, sk_{p_{m,0}}, \cdots, sk_{p_{1,u-1}}, \cdots, sk_{p_{m,u-1}}$ cannot be used.

are revoked. To revoke the privileges of particular service providers, the broadcaster adds their temporal identities to the end of the *CRL*. For example, to revoke the privileges of service providers p_i and p_j whose temporal identities are $cr_i = u \times n + p_i$ and $cr_j = u \times n + p_j$ ($u \times n < cr_i < cr_j \leq u \times n + m$) after u times' update, the broadcaster sets

$$CRL = \{u \times n, cr_i, cr_j\},$$

adds p_i and p_j to the $UPL = \{p_a, p_b, \cdots, p_h\}$, and creates a new

$$UPL = \{p_a, p_b, \cdots, p_h, p_i, p_j\}.$$

The broadcaster does not transmit partial keys to them during the next signing key update so that they cannot update their signing keys. Therefore, requirement (3) is satisfied. Moreover, as shown in Fig. 5, a signing key can easily be updated since the broadcaster only has to distribute a partial key to a service provider online and does not have to distribute a new signing key to a service provider offline. Only when the providers in the UPL want to restart their services, their partial or signing keys are distributed offline. It would be natural that the providers want to continue their services and do not want to be included in a *CRL* nor a UPL, and they try to manage their keys seriously. Thus, the number of such restarts is small and requirement (4) is satisfied.

In the above generic construction, the broadcaster runs $\mathsf{Upd}^*(msk,\ 0,\ p_i)$ to create a partial key sk'_{0,p_i} and $\mathsf{Upd}(0,\ p_i,\ sk_0,\ sk'_{0,p_i})$ to create a signing key sk_{p_i}. If the KIS scheme can deal with only sequential signing key updates, the broadcaster runs $\mathsf{Upd}^*(msk,\ p_{i-1},\ p_i)$ to create a partial key sk'_{p_{i-1},p_i} and $\mathsf{Upd}(p_{i-1},\ p_i,\ sk_{p_{i-1}},\ sk'_{p_{i-1},p_i})$ to create a signing key sk_{p_i}.

When giving a concrete instantiation of the protocol from a concrete KIS scheme by using the above generic construction, the security of the protocol depends on that of the KIS scheme. For example, the KIS schemes, that are developed by using generic construction from digital signature schemes in [4], are not secure against $sk'_{p_{i,u-1},p_{i,u}}$ exposure. That is, when the adversary gets $sk'_{p_{i,u-1},p_{i,u}}$, it can generate $sk_{p_{i,u}}$ even if the adversary does not have $sk_{p_{i,u-1}}$. On the other hand, the KIS scheme described in Sect. 4 in the same literature [4] is secure against the $sk'_{p_{i,u-1},p_{i,u}}$ exposure. These properties affect to the security of the protocols and the latter KIS scheme makes the protocol secure against $sk'_{p_{i,u-1},p_{i,u}}$ exposure.

In both of the generic constructions described in Sects. 4.3 and 4.4, the broadcaster distributes the verification key and the *CRL* to users through secure broadcast channels in order to prevent them from being modified by an adversary. However, the construction from KIS schemes is more efficient in revocation than that of the construction from IBS schemes since the structure of the *CRL* is simpler than that of the latter construction. In addition, the former construction has more secure signing key update than the latter construction since new signing keys are not sent to user terminals as they are. The former construction uses offline communication when the revoked service providers restart their services, and it is as secure as the latter construction. The above observations indicate that both generic constructions are suitable for hybrid services offered through broadcasting and communications networks, but that the construction from KIS schemes is more suitable than that from IBS.

5. Practical Protocol

As shown in Sect. 4, the generic construction of an application authentication protocol from a KIS scheme satisfies all of the requirements in Sect. 3, and is more efficient and more secure than that from an IBS scheme. Here, we propose a practical application authentication protocol for hybrid services offered through broadcasting and communications networks. Our protocol applies the KIS scheme proposed by Ohtake et al. [9].

5.1 Algorithms of Key-Insulated Signature Scheme [9]

The KIS scheme proposed by Ohtake et al. [9] is efficient in terms of the key size, signature size, and computational cost. The algorithms of the scheme are as follows:

Gen: Let $\mathbb{G}_q$ be a cyclic group of prime order q. Randomly select $g \in \mathbb{G}_q$, where g is a generator of $\mathbb{G}_q$. Then, randomly select $msk_x \in Z_q$ and $msk_{x'} \in Z_q$. The master key msk_x is stored in a secure device, and $msk_{x'}$ is managed by a signer. Calculate $y = g^{msk_x}$ and $y' = g^{msk_{x'}}$, and publish the verification key $vk = \langle q, g, y, y', G(\cdot,\cdot), H(\cdot,\cdot,\cdot,\cdot)\rangle$. Here, G and H are hash functions, where $G : \mathbb{G}_q \times \{0,1\}^* \to Z_q$ and $H : \mathbb{G}_q \times \mathbb{G}_q \times \{0,1\}^* \times \{0,1\}^* \to Z_q$.

Upd^*: Randomly select $r_t \in Z_q$ from a secure device, and calculate $v_t = g^{r_t}$. Then, calculate $c_t = G(v_t, t)$ using the inputted time period t, and obtain a partial key $sk'_t = c_t r_t + msk_x \bmod q$ using the inputted master key msk_x. sk'_t, v_t, and t are transmitted to the signer.

Upd: The signer obtains the signing key $sk_t = sk'_t + msk_{x'} \bmod q$ for a time period t by using sk'_t and the inputted $msk_{x'}$.

Sign: The signer randomly selects $r_s \in Z_q$ and calculates $v_s = g^{r_s}$. Then, the signer calculates $c_s = H(v_t, v_s, t, M)$ and $\sigma = c_s r_s + sk_t \bmod q$ by using the inputted message M, t, the signing key sk_t, and v_t. Finally, the signer transmits M, (σ, c_s, v_t), and t to a verifier.

Vrfy: Using the inputted y, y' M, (σ, c_s, v_t), and t, the verifier calculates $c_t = G(v_t, t)$. If the following equation holds, it returns $b = 1$; otherwise, it returns $b = 0$.

$$c_s = H(v_t, (g^{\sigma}(v_t^{c_t} yy')^{-1})^{1/c_s}, t, M)$$

The above scheme is a *strong* KIS scheme, which is secure against either signing key leakage or master key leakage. That is, an adversary cannot create a new signing key unless both the master keys msk_x and $msk_{x'}$ (or a signing key sk_t) are leaked. In addition, an adversary given $sk'_{p_{i,u-1},p_{i,u}}$ but not $sk_{p_{i,u-1}}$ is unable to create $sk_{p_{i,u}}$ and to forge a signature on the period after the uth update.

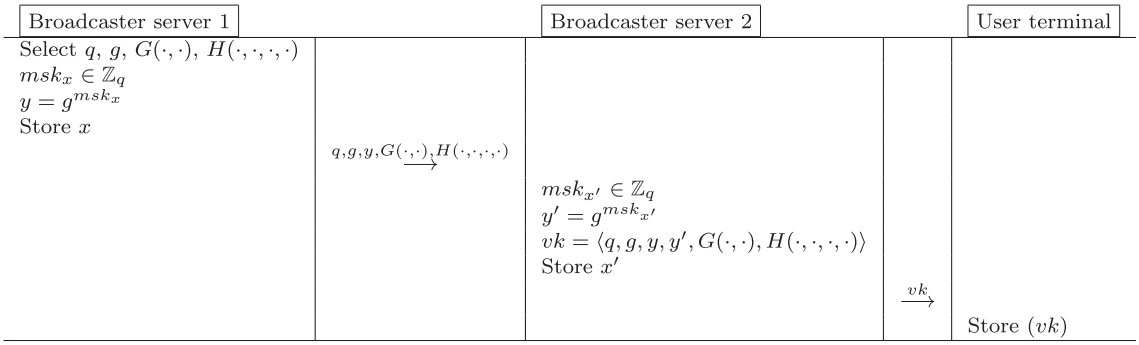

Fig. 6 Proposed protocol (initial settings)

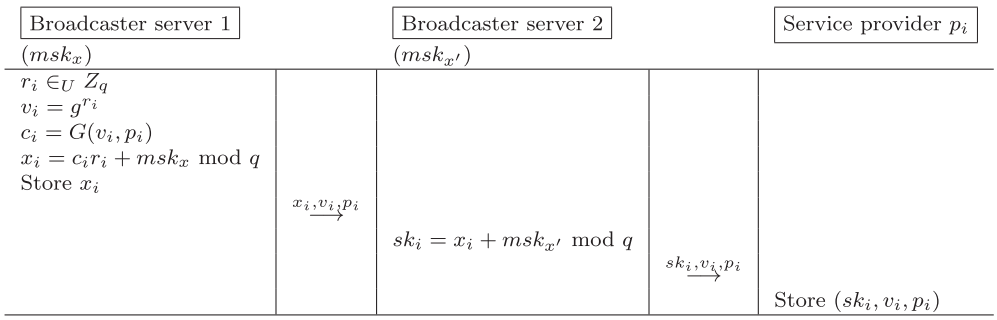

Fig. 7 Proposed protocol (initial issuance of signing key)

5.2 Proposed Protocol

We propose an application authentication protocol by using the algorithms of the KIS scheme in Sect. 5.1.

(1) Initial Settings

Figure 6 shows the procedure for making the initial settings before the start of services. The broadcaster server 1 executes the Gen algorithm of the KIS and generates q, g, $G(\cdot,\cdot)$, and $H(\cdot,\cdot,\cdot,\cdot)$. It then selects $msk_x \in \mathbb{Z}_q$ and calculates $y = g^{msk_x}$. It transmits q, g, y, $G(\cdot,\cdot)$, and $H(\cdot,\cdot,\cdot,\cdot)$ to server 2. Server 2 selects $msk_{x'} \in \mathbb{Z}_q$ and calculates $y' = g^{msk_{x'}}$. It then generates $vk = \langle q, g, y, y', G(\cdot,\cdot), H(\cdot,\cdot,\cdot,\cdot)\rangle$. It provides vk to all user terminals.

(2) Signing key issue.

Figure 7 shows the procedure of issuing the signing key for a service provider p_i, where p_i $(i = 1, 2, \ldots, m)$ is an identity for the service provider. The broadcaster has two servers. Broadcaster server 1 securely manages the master key msk_x, and broadcaster server 2 securely manages the key $msk_{x'}$. Broadcaster server 1 randomly selects $r_i \in Z_q$ and calculates $v_i = g^{r_i}$. Then, it obtains $c_i = G(v_i, p_i)$ by using p_i and obtains a partial key $x_i = c_i r_i + msk_x \bmod q$ by using the master key msk_x. Broadcaster server 1 stores x_i for the service provider p_i and transmits x_i, v_i, and p_i to broadcaster server 2. Broadcaster server 2 calculates the signing key $sk_{p_{i,0}} = x_i + msk_{x'} \bmod q$ by using $msk_{x'}$ and transmits sk_{p_i}, v_i, and p_i to a service provider p_i. The service provider stores them.

(3) Signing key update.

Figure 8 shows the procedure of updating the signing key for a service provider p_i. The signing key update is performed for all of the service providers except for the providers that are included in the UPL. Let $p_{i,u} = u \times n + p_i$ be the temporal identity of the service provider p_i after the uth signing key update. Broadcaster server 1 randomly selects $r'_i \in Z_q$ and calculates $v'_i = g^{r'_i}$. Then, it obtains $c'_i = G(v'_i, p_{i,u})$ by using $p_{i,u}$ and obtains $x'_i = c'_i r'_i + msk_x \bmod q$ by using the master key msk_x. Broadcaster server 1 calculates a partial key $\Delta x'_i = x'_i - x_i \bmod q$ and stores x'_i by overwriting x_i. Then it transmits $\Delta x'_i$, v'_i, and $p_{i,u}$ to the service provider p_i. The service provider p_i obtains a new signing key $sk_{p_{i,u}} = sk_{p_{i,u-1}} + \Delta x'_i \bmod q$ by using the current signing key $sk_{p_{i,u}}$ and stores $(sk_{p_{i,u}}, v'_i, p_{i,u})$ by overwriting $(sk_{p_{i,u-1}}, v_i, p_{i,u-1})$. The above process enables the signing key for service provider p_i to be updated as follows: $sk_{p_{i,0}} \to sk_{p_{i,1}} \to sk_{p_{i,2}} \to \cdots$.

It should be noted that, when the service provider in the UPL would like to restart the service, the provider has to obtain a new signing key. In this case, the broadcaster sends a partial key or a new signing key through offline communication.

In addition, an adversary given $\Delta x'_i$ but not $sk_{p_{i,u-1}}$ is unable to create $sk_{p_{i,u}}$ and to forge a signature on the period after the uth update. This property depends on the security of the used KIS scheme [9].

Broadcaster server 1 $(msk_x,\ x_i,\ UPL)$		Service provider p_i $(sk_{p_{i,u-1}})$
$p_{i,u} = u \times n + p_i$ $r'_i \in_U Z_q$ $v'_i = g^{r'_i}$ $c'_i = G(v'_i, p_{i,u})$ $x'_i = c'_i r'_i + msk_x \bmod q$ $\Delta x'_i = x'_i - x_i \bmod q$ Store x'_i overwriting x_i	$\xrightarrow{\Delta x'_i, v'_i, p_{i,u}}$	$sk_{p_{i,u}} = sk_{p_{i,u-1}} + \Delta x'_i \bmod q$ Store $(sk_{i_u}, v'_i, p_{i,u})$ overwriting $(sk_{p_{i,u-1}}, v_i, p_{i,u-1})$

Fig. 8 Proposed protocol (the uth update of signing key ($sk_{p_{i,u-1}} \rightarrow sk_{p_{i,u}}$))

Service provider p_i $(sk_{p_{i,u}}, v_i, p_{i,u})$		User terminal (vk, CRL)
$r_s \in_U Z_q$ $v_s = g^{r_s}$ $c_s = H(v_i, v_s, p_{i,u}, M)$ $\sigma_i = c_s r_s + sk_{p_{i,u}} \bmod q$	$\xrightarrow{M, (\sigma_i, c_s, v_i), p_{i,u}}$	Output error message if $p_{i,u}$ is included in the CRL $c' = G(v_i, p_{i,u})$ $c_s \stackrel{?}{=} H(v_i, (g^{\sigma_i}(v_i^{c'} yy')^{-1})^{1/c_s}, p_{i,u}, M)$

Fig. 9 Proposed protocol (authenticate application)

(4) Application authentication.

Figure 9 shows the procedure of application authentication when service provider p_i distributes an application to a user terminal. The service provider p_i randomly selects $r_s \in Z_q$ and calculates $v_s = g^{r_s}$. Then, it obtains $c_s = H(v_i, v_s, p_{i,u}, M)$ and $\sigma_i = c_s r_s + sk_{p_{i,u}} \bmod q$ by using an application M, $p_{i,u}$, $sk_{p_{i,u}}$, and v_i, and it transmits M, (σ_i, c_s, v_i), and $p_{i,u}$ to the user terminal. The user terminal gets a *CRL* from broadcaster server 1. If $p_{i,u}$ is included in the *CRL*, the terminal outputs an error message. Otherwise, it calculates $c' = G(v_i, p_{i,u})$. If the following equation holds, the authentication is successful.

$$c_s = H(v_i, (g^{\sigma_i}(v_i^{c'} yy')^{-1})^{1/c_s}, p_{i,u}, M)$$

(5) Add service provider.

To add a trusted service provider, the broadcaster assigns p_i $(m < p_i \le n)$ to the identity of the new service provider and issues its signing key through the *signing key issue* procedure depicted in Fig. 7.

(6) Revoke privileges of service provider.

Broadcaster server 1 distributes a *CRL* including the temporal identities of the service providers whose privileges the broadcaster wants to revoke to all user terminals. The way of setting the *CRL* is as follows:

- *Revoke privileges of all service providers*
  If the broadcaster wants to revoke the privileges of all service providers at once, it sets the maximum number of the temporal identities to the first element of the *CRL*. For example, for all of the identities $1, 2, \ldots, R$, the broadcaster may set
  $$CRL = \{R\}.$$
- *Revoke privileges of a particular provider*
  To revoke the privileges of a particular provider, the broadcaster adds the temporal identities to end of the *CRL*, and it adds their identities to end of the *UPL*. For example, to revoke the privileges of service providers p_i and p_j whose temporal identities are $cr_i = u \times n + p_i$ and $cr_j = u \times n + p_j$ $(u \times n < cr_i < cr_j \le u \times n + m)$ after u times' update, the broadcaster sets
  $$CRL = \{u \times n, cr_i, cr_j\}$$
  $$UPL = \{p_a, p_b, \cdots, p_h, p_i, p_j\}.$$

It should be noted that the *UPL* is not transmitted to the user terminals and is hold by the broadcaster. That is, it does not affect the storage size nor the CPU cost of the terminals.

(7) Restart services from revocation.

The providers included in the *UPL* cannot obtain their partial keys and cannot update their signing keys. It results that these providers cannot generate valid signatures for their services. In order to restart their services, they have to get new valid signing keys, and for this purpose, the broadcaster transmits their new signing keys or their partial keys offline.

There are many kinds of reasons to put p_i into the *UPL*. One of the reasons and the worst reason among them is the

exposure of its signing key. In this case, the partial key cannot be transmitted online (through insecure channel) since the adversary that has the exposed signing key can get the partial key by eavesdropping the communication between the broadcaster and the service provider and can generate a new valid signing key. Thus, this transmission should be offline.

After this transmission, the service provider updates its signing key (or get it directly), and then, the broadcaster removes p_i from the $UPL = \{p_a, p_b, \cdots, p_h, p_i, p_j\}$ and creates a new $UPL = \{p_a, p_b, \cdots, p_h, p_j\}$.

In the *signing key update* process, the broadcaster updates the signing key for a service provider p_i as follows: $sk_{p_{i,0}} \rightarrow sk_{p_{i,1}} \rightarrow sk_{p_{i,2}} \rightarrow \cdots$. This is because the broadcaster uses a temporal identity $p_{i,u} = u \times n + p_i$ after the uth signing key update, which is constructed from two-dimensional elements: the number of update $u = 1, 2, \ldots$ and the real identity $1 \leq p_i \leq m$.

In the proposed protocol, the number of service providers is limited by the parameter n. However, this is not a fatal weakness because the broadcaster can set n to a sufficiently large number that will let the number of service providers grow in the future.

5.3 Security of Proposed Protocol

Our protocol is based on the KIS scheme proposed by Ohtake et al. [9], which is provably secure under the discrete logarithm assumption (See [9]). Therefore, here, we will only discuss the security of the proposed protocol against key leakage within the system.

Our protocol is secure because the broadcaster uses two servers. Let us consider the case in which the master key msk_x is leaked from broadcaster server 1. In this case, the keys x_i $(1 \leq i \leq m)$, managed by the same server, may be leaked. However, another master key $msk_{x'}$ is managed by broadcaster server 2, so an adversary cannot create a signing key $sk_{p_i} = x_i + msk_{x'}$ unless $msk_{x'}$ is leaked simultaneously. An adversary can create x_i, x'_i, and their difference $\Delta x'_i$. However, since the signing key $sk_{p_{i,u-1}}$ is managed by the service provider p_i, the adversary cannot create a signing key $sk_{p_{i,u}} = sk_{p_{i,u-1}} + \Delta x'_i$ unless $sk_{p_{i,u-1}}$ is leaked as well.

Now let us consider the case in which key $msk_{x'}$ is leaked from broadcaster server 2. In this case, an adversary cannot create x_i unless msk_x is leaked from broadcaster server 1 at the same time. Therefore, the adversary cannot create a signing key $sk_{p_i} = x_i + msk_{x'}$.

Let us consider the case in which the signing key $sk_{p_{i,u}}$ is leaked from service provider p_i. In this case, the broadcaster revokes the signing key $sk_{p_{i,u}}$ by using a *CRL* as described in Sect. 5.2.

Finally, let us consider the update of a signing key $sk_{p_{i,u-1}}$ of a provider that is in a UPL. We described in the above that p_i is revoked when the identity p_i is in the UPL. The partial key $\Delta x'_i$ is necessary to update $sk_{p_{i,u-1}}$. However, it is not generated when p_i is in the UPL. Moreover, the other partial keys $\Delta x'_j$ $(j \neq i)$ are different from $\Delta x'_i$. Even if the adversary eavesdrops the communication between the broadcaster and service providers and gets all $\Delta x'_j$s, it cannot generate a valid signing key $sk_{p_{i,u}}$. Concretely, even if the adversary got $sk_{p_{i,u-1}}$ and all $\Delta x'_j$s, it cannot generate $\Delta x'_i = x'_i - x_i$. That is, it does not know $x'_i = c'_i r'_i + msk_x$, x_i and msk_x, and it cannot generate a correct $\Delta x'_i$. More concretely, it cannot generate msk_x from $sk_{p_{i,u-1}}$ and all $\Delta x'_j$s. Hence, the adversary fails to update $sk_{p_{i,u-1}}$ and to generate $sk_{p_{i,u}}$. Moreover, an adversary given $\Delta x'_i$ but not $sk_{p_{i,u-1}}$ is unable to create $sk_{p_{i,u}}$ and to forge a signature on the period after the uth update. This property depends on the security of the used KIS scheme [9].

In Fig. 7, broadcaster server 2 must issue a signing key sk_{p_i} offline to the service provider p_i. However, in Fig. 8, broadcaster server 1 can transmit a partial key $\Delta x'_i$ online to the service provider since $\Delta x'_i$ has no information about the signing key. The strong point here is that the signing key update can be performed online (See Fig. 5) except for the restart from revocation. In contrast, in an application authentication protocol using an IBS (Fig. 3), the broadcaster must transmit the new signing key to the service provider when updating the signing key. Therefore, it is inconvenient that its signing key update is performed always offline.

5.4 Server Management

The proposed protocol needs at least two servers in a broadcaster, and the following conditions should be met:

i. Administrator: The servers are managed by distinct administrators.
ii. Location: The servers are located in distinct rooms.
iii. Network: The servers are only connected when they need to be.
iv. OS: The servers use distinct operation systems.

More concretely, when the servers are managed by one administrator, the administrator can generate all the signing keys at will and the system is not secure against potentially illegal actions of the administrator. Therefore, condition i is necessary. Condition ii is required in order for distinct administrators to manage servers. In addition, such a measure can decrease the chances for persons other than the administrators to operate simultaneously both the servers, obtain signing keys, and steal the two servers. The servers do not always need to be connected physically or virtually. Their roles are separated and, a temporal connection is necessary only when x_i, v_i and p_i are transmitted. If the servers are always connected, an attack on one server may affect the other server. At least, the probability of such an affect increases, and the security established by server separation consequently deteriorates. Hence, condition iii is necessary. Moreover, it is preferable that the two servers use distinct OSs, distinct security policies, distinct security software, etc. These measures are effective against cyber-attacks and this is why condition iv is required.

Actually, almost all broadcasters that supply services

Table 1 Comparison of application authentication protocols: n denotes the number of service providers. 'Ordinary' denotes the signature schemes used in the current PKI, e.g. DSA, ECDSA, RSA-PSS, RSASSA schemes. r denotes the number of revoked service providers' identities that are in a *CRL*. u denotes the update times of all signing keys.

		PKI1	PKI2	PKI3	IBS	OURS
Storage for verification key(s)		$O(n)$	0	$O(1)$	$O(1)$	$O(1)$
Verification cost	Minimum #Verification	1	2	2	1	1
	Signature scheme	Ordinary	Ordinary	Ordinary	IBS	KIS
	Network cost	0	1	0	0	0
Revocation efficiency	Certificate	necessary	necessary	necessary	unnecessary	unnecessary
	CRL size	$O(r)$	$O(r)$	$O(r)$	$O(u \times n + r)$	$O(r)$
Ease of signing key update	Update	offline	offline	offline	offline	online
	Restart	offline	offline	offline	offline	offline

through the air have some rooms available and it would be easy for them to locate two servers in different places. However, the network between the servers is always connected. Its physical or virtual disconnection is possible but requires great care. In addition, it is difficult to have different administrators manage the servers, because most broadcasters outsource the job to one company. Outsourcing the job to two companies, for instance, may double the cost. That is, there is a trade-off relationship between cost and security.

Finally, if the proposed protocol is introduced at the present moment, there would be difficulties in using multiple servers. However, the use of multiple servers and multiple administrators will be inevitable because of the growing need to construct more secure system, and the broadcasters have to consider the trade-off relationship in detail.

5.5 Performance Evaluation

Table 1 compares application authentication protocols according to requirements (1)-(4) in terms of the amount of storage occupied by their verification keys, the verification cost, revocation efficiency, and ease of signing key update.

"PKI1" denotes the application authentication protocol in Model 1-1 or Model 2-1, "PKI2" denotes that in Model 1-2 or Model 2-2, and "PKI3" denotes that in Model 1-3 or Model 2-3. "IBS" denotes an application authentication protocol using an IBS scheme, and "OURS" denotes the application authentication protocol described in Sect. 5.2. IBS and OURS can be applied to Model 2 in Fig. 2.

'Storage for verification keys' means the amount of storage the verification keys take up in a terminal. It is related to requirement (1) and the lesser amount is acceptable. 'Verification cost' is related to requirement (2) and we compare the protocols in sub-terms of the minimum number of verifications, signature schemes, and network cost. 'Minimum #verification' means the minimum number of verifications when a terminal receives an application, and the smaller number is acceptable. 'Signature scheme' means the signature scheme used in the model[†]. 'Network cost' means the minimum number of communication[††] through networks for the authentication between the terminal and some entities, and the smaller number is acceptable. 'Revocation efficiency' is related to requirement (3) and we compare the protocols in sub-terms of necessity of certificate and size of a *CRL*. 'necessary' and 'unnecessary' on the 'Certificate' row shows whether certificates of service providers are used to verify the validity of verification keys or not. It is desirable not to use the certificate since the validity check of corresponding verification key is unnecessary. '*CRL* size' means the number of elements in a *CRL* and shows the number of revoked service providers' identities or temporal identities. The size affects the CPU cost of revocation and the smaller number is acceptable. 'online' and 'offline' on the 'Ease of signing key update' rows show whether the broadcaster updates the signing keys of the service providers online or offline. 'Update' means the ordinary update when the temporal identity is not included in the *CRL* nor when the identity is not included in a UPL and 'Restart' means the update when the identity is included in the UPL. The restart process is basically the same as that of signing key issuance and is offline. These are related to requirement (4) and updating online is more desirable.

PKI1 protocol employs ordinary signature schemes, e.g. DSA, ECDSA, RSA-PSS, and RSASSA. In the protocol, the user terminal requires all of the verification keys and certificates of the service providers in order to verify a signature. Therefore, the amount of storage used by the verification keys depends on the number of service providers. This protocol requires only one verification that is for the signature on the application, and its CPU cost for the verification is smaller than those of IBS and KIS. Furthermore, it does not communicate with any entity in this verification since all necessary certificates are stored in the terminal beforehand. If the broadcaster wants to revoke the privileges of a service provider whose applications caused an undesirable action to occur, it issues a *CRL* including the identity of the provider. The *CRL* is the same as that of the current PKI and the number of identities in the *CRL* equals that of revoked service providers r. When a user terminal verifies a signature on an application, it confirms the validity of the verification key by using the certificate and verifies the signature by using the verification key. It also confirms that the identity of the verification key is not in the *CRL*. Here, the identities of the verification key and certificate of a service provider can be the same as that of the provider without

[†]We assume that the signature schemes used in Model 1 are used in the current PKI.

[††]We define that one communication is a sequence of communications from asking the other entity to verify a signature to obtaining its result.

loss of generality and we use an identical identity for these three. The processes for revocation are the same as those of the current PKI. To update the signing key of a service provider or to restart the service of the service provider that is included in the *CRL*, the broadcaster reissues a certificate. This process must be performed offline.

PKI2 protocol employs ordinary signature schemes. In the protocol, a user terminal does not store any verification key, and thus, the amount of storage taken up by the verification keys is independent of the number of providers. The minimum number of verifications is two: one for the signature on the application, the other for the signature on the certificate. Its CPU cost for the verification would be smaller than those of IBS and KIS. However, when a user terminal verifies a signature on an application, it has to communicate with broadcasters to verify the certificate attached to the signature. The minimum number of communication is one according to that of the certificates. After the communication, the same processes of PKI are performed, and hence, the number of identities in a *CRL* is r. Updating the signing key of a service provider and reissuing the signing key to restart the service of the service provider that is included in the *CRL* must be performed offline and is not easy.

PKI3 protocol employs ordinary signature schemes. In the protocol, a user terminal holds only the broadcasters' verification keys, and thus, the amount of storage needed for the verification keys is small. The minimum number of verifications is two: one for the signature on the application, the other for the signature on the certificate. To make these verifications, the user terminal does not communicate with any entity. The processes to verify signatures on an application are as same as that of PKI1. Updating the signing key of a service provider and reissuing the signing key to restart the service of the service provider that is included in a *CRL* must be performed offline and is not easy.

IBS protocol employs IBS scheme. In the protocols, a user terminal uses only one verification key for verifying the signature on the application and it does not use any certificate. The minimum number of verifications is thus one, but the terminal does not require any communication with other entities. Therefore, the amount of storage occupied by the verification key is independent of the number of service providers. If a broadcaster wants to revoke the privileges of certain service providers, it adds their identities to a *CRL*. If it does not need to update signing keys, the number of identities in the *CRL* equals that of revoked service providers r. However, it would be necessary to update all signing keys periodically in order to securely manage the system, and in this case, the number of identities in the *CRL* is $u \times n + r$. To update the signing key of a service provider or to restart the service of the service provider that is included in the *CRL*, the broadcaster reissues a signing key corresponding to the new identity offline.

Our protocol employs KIS scheme. In the protocol, a user terminal uses only one verification key to verify the signature and it does not need any certificate. Therefore, the amount of storage occupied by the verification key is independent of the number of service providers. Moreover, the minimum number of verifications is one. Verification of the signature on the application is necessary, but the terminal does not require any communication with other entities. If the broadcaster wants to revoke the privileges of service providers, it puts the temporal identities in a *CRL* and their identities in a *UPL*. Moreover, the temporal identity of a service provider is always a sequential number and the broadcaster can revoke them all at once when it updates all signing keys. The description of the *CRL* is simpler than in IBS, as described in Sects. 4.4 and 5.2. When the provider whose temporal identities are cr_i and cr_j are revoked after u times' signing key updates, the *CRL* of our protocol is $\{u \times n, cr_i, cr_j\}$ and that of IBS is $\{1, 2, \cdots, u \times n, cr_i, cr_j\}$. The numbers of identities in the *CRL*s are $r+1$ and $u \times n + r$, respectively. It should be noted that the *UPL* is not transmitted to user terminals. To update the signing key for a service provider, the broadcaster transmits the partial key used for the signing key update to the service provider. The partial key has no information about the signing key. That is, the broadcaster can transmit the partial key online without using a secret channel. However, when the service provider included in the *UPL* would like to restart its service, it has to get a new signing key or a partial key offline. This process requires the same cost as that of the IBS model. It would be natural that the providers want to continue their services and do not want to be included in a *CRL* nor a *UPL*, and thus, the number of such restarts is small. This means that the number of service providers that need to restart from revocation is much less than that of providers that update their signing key ordinarily. That is, although there is no advantage of our model when the providers in the *UPL* restart their services, there is advantage of our model when the signing keys are updated ordinarily. Hence, our model can more easily update the signing key than all the other models.

The above discussion clearly shows that our protocol improves the efficiency of the conventional methods.

6. Conclusion

We proposed an application authentication protocol for hybrid services that make use of broadcasting and communications networks. We modified a KIS scheme and applied it to this protocol. In particular, we considered time periods in a KIS scheme to be a temporal identity of a service provider. That is, the service provider's identity is two-dimensional one consisting of the time period and its real identity. We showed that our protocol is secure and practical.

Currently, hybrid services are being developed but the number of applications is small. In addition, only a small number of companies are developing applications and application authentication is as yet unnecessary. However, third parties are allowed to develop applications, and as the number of parties increases, application authentication will become mandatory. Broadcasters may have to take security precautions against malicious applications made by a lot of service providers [5]. By using our protocol in such situ-

ations, users can securely receive hybrid services through broadcasting and communications networks.

Acknowledgments

We would like to thank Kinji Matsumura for his comments on the applications of hybrid services. We would also like to thank Masaru Takechi for his valuable comments on the ability of CPUs in terminals, receivers, and set-top boxes. We would also like to thank Goichiro Hanaoka for a lot of discussions and suggestions on the signature schemes. We would also like to thank the editors and anonymous reviewers for their invaluable comments. Owing to their comments, we could improve this paper.

References

[1] A. Baba, K. Matsumura, S. Mitsuya, M. Takechi, H. Fujisawa, H. Hamada, S. Sunasaki, and H. Katoh, "Seamless, Synchronous, and Supportive: Welcome to Hybridcast - An advanced hybrid broadcast and broadband system," IEEE Consum. Electron. Mag., vol.1, no.2, pp.43–52, 2012.
[2] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," Proc. of EUROCRYPT'04, LNCS 3027, pp.223–238, Springer-Verlag, 2004.
[3] D. Boneh and M.K. Franklin, "Identity-based encryption from the weil pairing," Proc. of CRYPTO'01, LNCS 2139, pp.213–229, Springer-Verlag, 2001.
[4] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong Key-Insulated Signature Schemes," Proc. of PKC'03, LNCS 2567, pp.130–144, Springer-Verlag, 2003.
[5] Y. Hironaka, H. Ohmata, G. Ohtake, K. Otsuki, M. Takechi, K. Kai, and K. Majima, "A Prototype System of Application Distribution Management for Integrated Broadcasting Communications Services," IPSJ SIG Technical Report, vol.2014-AVM-83, no.32, pp.1–6, 2013 (in Japanese).
[6] C. Gentry, "Practical identity-based encryption without random oracles," Proc. of EUROCRYPT'06, LNCS 4004, pp.445–464, Springer-Verlag, 2006.
[7] N. González-Deleito, O. Markowitch, and E. Dall'Olio, "A New Key-Insulated Signature Scheme," Proc. of ICICS'04, LNCS 3269, pp.465–479, Springer-Verlag, 2004.
[8] D. Lee, M. Lee, and D. Kang, "OHTV(open hybrid TV) service platform based on terrestrial DTV," Proc. of ICACT'10, IEEE Press, pp.399–402, 2010.
[9] G. Ohtake, G. Hanaoka, and K. Ogawa, "Efficient Provider Authentication for Bidirectional Broadcasting Service," IEICE Trans. Fundamentals, vol.E93-A, no.6, pp.1039–1051, 2010.
[10] G. Ohtake and K. Ogawa, "Application authentication for hybrid services of broadcasting and communications networks," Proc. of WISA'11, LNCS 7115, pp.171–186, Springer-Verlag, 2011.
[11] A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," Proc. of CRYPTO'84, LNCS 196, pp.47–53, Springer-Verlag, 1984.
[12] B. Waters, "Efficient identity-based encryption without random oracles," Proc. of EUROCRYPT'05, LNCS 3494, pp.114–127, Springer-Verlag, 2005.
[13] http://www.hbbtv.org/
[14] HbbTV, "ETSI TS 102 796 V1.2.1 Errata 2," http://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.03.01_60/ts_102796v010301p.pdf
[15] http://www.hulu.com/
[16] http://www.nhk.or.jp/hybridcast/online/
[17] IPTV Forum Japan, "Integrated Broadcast-Broadband System Specification," http://www.iptvforum.jp/download/docs/files/HybridcastSystem20e.pdf
[18] IPTV Forum Japan, "HTML5 Browser Specification," http://www.iptvforum.jp/download/docs/files/HTML5Browser21e.pdf
[19] http://www.urlkor.com/w.ohtv.kr (in Korean)
[20] http://www.cableplus.jp/smart-tv-box/
[21] http://www.synccast.jp/
[22] http://www.youview.com/

Kazuto Ogawa received the B.E. and Ph.D. degrees from the University of Tokyo, Tokyo, Japan in 1987 and 2008, respectively. He joined NHK (Japan Broadcasting Corporation) in 1987. He has mainly engaged in research and development on video image processing systems and digital content rights management systems. He is currently a research engineer of NHK Science and Technical Research Laboratories.

Go Ohtake received the B.E. and M.E. degrees from Tokyo Institute of Technology in 1999 and 2001, respectively. He joined NHK (Japan Broadcasting Corporation) in 2001 and received the Ph.D. degree from Institute of Information Security in 2009. He is currently a research engineer of NHK Science and Technical Research Laboratories and a visiting researcher at University of Calgary. His research interests include public key cryptography and its application for copyright protection and privacy preserving.