IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 31 August 2014; revised 7 December 2014 and 18 January 2015; accepted 19 January 2015.
Date of publication 17 February 2015; date of current version 26 February 2016.

Digital Object Identifier 10.1109/TETC.2015.2400131

Secure and Anonymous Communication Technique: Formal Model and Its Prototype Implementation

KEITA EMURA[1], AKIRA KANAOKA[2], SATOSHI OHTA[1], KAZUMASA OMOTE[3], AND TAKESHI TAKAHASHI[1], (Member, IEEE)

[1]Network Security Research Institute, National Institute of Information Communications Technology, Tokyo 184-8795 Japan
[2]Department of Information Science, Toho University, Chiba 274-8510, Japan
[3]Japan Advanced Institute of Science and Technology, Nomi 923-1211, Japan
CORRESPONDING AUTHOR: K. EMURA (k-emura@nict.go.jp)

ABSTRACT Both anonymity and end-to-end encryption are recognized as important properties in privacy-preserving communication. However, secure and anonymous communication protocol that requires both anonymity and end-to-end encryption cannot be realized through a simple combination of current anonymous communication protocols and public key infrastructure (PKI). Indeed, the current PKI contradicts anonymity because the certificate for a user's public key identifies the user. Moreover, we believe that anonymous communication channels should have certain authentication mechanisms because such a channel could incubate criminal communication. To cope with this issue, we propose a secure and anonymous communication protocol by employing identity-based encryption for encrypting packets without sacrificing anonymity, and group signature for anonymous user authentication. Communication occurs in the protocol through proxy entities that conceal user IP addresses from service providers (SPs). We also introduce a proof-of-concept implementation to demonstrate the protocol's feasibility and analyze its performance. Finally, we conclude that the protocol realizes secure and anonymous communications between users and SPs with practical performance.

INDEX TERMS Anonymous communication, anonymous authentication, secure channel, identity-based encryption, group signature.

I. INTRODUCTION

Anonymity[1] is an important aspect of privacy, and systems that provide services to ensure user anonymity are currently a topic of keen interest. Such systems can provide services to users without revealing their identity. With regard to the latter, a number of studies have been reported on [1], and many of those studies use cryptography as the important building block for constructing the systems; however, these need further improvement before they can be used for actual services.

A. RESEARCH BACKGROUND

Several cryptographic primitives that can provide anonymity have been proposed. Among these is group signature [2], which allows signers to prove anonymously the validity of signatures. A group manager (GM) with a pair of a group public key, gpk, and a master secret key, msk, issues a secret signing key, sk_i, to a user U_i that computes a group signature, σ (on certain messages), using sk_i. No user-dependent value is required in the verification phase; a verifier verifies σ using only the corresponding gpk. However, these approaches alone cannot guarantee anonymity when applied to online communication. For instance, let a signer compute a group signature and *send* it to a verifier. The verifier can anonymously verify the signature's validity. However, there is a question of how to send anonymously the group signature to the verifier. Usually, a source IP address is included in a packet that reveals the identity of the sender thus user anonymity is already infringed. The situation remains the same regardless of the primitives we implement provided that direct communication between a sender (signer, prover, etc.) and a receiver (verifier, etc.) is required.

User IP address is naturally visible in the IP packets sent from the user, and it cannot simply be erased or forged to allow bi-directional communication. One approach for this

[1]This paper considers sender/prover anonymity and does not consider recipient anonymity.

is to use intermediate agents that send packets on behalf of the actual user terminal, and several such protocols have already been proposed [1], including Tor [3]. Nevertheless, another issue arises in the question of how to assure user legitimacy. We need to discern between legitimate and illegitimate users in order to restrict unauthorized access to a particular channel. One might believe that only end-to-end authentication is required, but it is difficult to authenticate users without identifying them. For instance, a server needs to send a response code to a user in basic authentication and the user needs to return a user ID and password. That is, the server needs to identify the user. Moreover, it seems difficult to send a certain message from the server to a user because the corresponding source IP address is generally required. Authentication by an intermediate agent (as in Tor [3]) might be a solution to these problems. The agent can authenticate a user and can hide the user's source IP address from the server. Nonetheless, we still need to know how the server can authenticate end users directly.

A simple approach to solve anonymous authentication problems is simply to combine both cryptographic primitives and anonymous communication protocols as follows. Let a user compute an anonymously-authenticated token (e.g., group signature), and send it to a server via an anonymous channel (e.g., using Tor). Then, the server can directly authenticate the user without compromising anonymity. However, another problem arises here: how to establish a secure channel (i.e., flowed data is encrypted). If the server utilizes a user public key (certified by a trusted Certificate Authority (CA) in a public key infrastructure (PKI)), it can identify the user because a certificate contains information on the key holder. The same problem arises even if symmetric key encryption is used. Assume that the server attempts to exchange a secret key with an end user. Because the server does not know the actual end user, it does not know its user public key for executing a key exchange protocol.

In summary, though both anonymity and end-to-end encryption are recognized as important properties in privacy-preserving communication, the current PKI contradicts anonymity, and it is highly desirable to construct a secure and anonymous communication protocol that achieves anonymity and end-to-end encryption simultaneously.

B. OUR CONTRIBUTION

In this paper, we propose a secure and anonymous communication protocol. The proposed protocol uses identity-based encryption (IBE) to encrypt content without identifying key holders.IBE can set arbitrary values on public keys; thus, it can allow a user to select a temporary ID for each session that the server can use as a public key. The protocol also uses group signature for anonymous user authentication. Communication in the protocol occurs through proxy entities that conceal user IP addresses from service providers (SPs).

This paper provides a framework for the proposed protocol, as well as a formal model and security definitions of the proposed protocol in a cryptographic manner; moreover, we prove that our system is secure in the cryptographic sense. From the perspective of efficiency, we demonstrate the needlessness of the group signature's open capability for our use, and then propose an open-free variant of the Furukawa-Imai group signature scheme [4]. This modification can reduce its signature size by 50% compared to the original scheme. Note that if someone needs to identify an illegitimate user, we can add such a mechanism without relying on cryptographic techniques; e.g., an IP address managed by proxy.

We demonstrate the feasibility and practicality of the proposed protocol by introducing our proof-of-concept implementation. The implementation uses the modified group signature scheme aforementioned and the Boneh-Franklin IBE scheme [5] for its underlying IBE scheme. Such implementation also uses a protocol-specific HTTP method in order to run the protocol over the Internet. We compare its performance to SSL communication, and conclude that the protocol realizes secure and anonymous communication between users and SPs with practical performance. Moreover, we show that the costs of cryptographic operations of our protocol are not dominant when the protocols are run over Tor networks.

As a remark, although communication among Tor routers is encrypted, this is not an end-to-end encryption. More precisely, a sender sequentially encrypts a content by using public keys of Tor routers and a *receiver*, and therefore the sender cannot encrypt the content without knowing the receiver public key. This situation contradicts anonymity. Whereas our protocol establishes end-to-end secure channels and therefore, no Tor routers can reveal content information in our protocol.

C. RELATED WORK

In general, the security of cryptographic primitives such as public key encryption or digital signature has to be proven mathematically. This provable security guarantees that no adversary exists unless the underlying complexity assumptions are broken. Recently, even secure ''systems'' that employ cryptographic primitives as their building blocks are required to be provably secure. Some examples are Transport Layer Security (TLS) [6], Kerberos [7], Single Sign on [8] and so on. Similarly to these systems, the security of our system can be proven mathematically.

Similar attempts to our approach do exist. Sudarsono et al. [9] considered an anonymous IEEE802.1X authentication system using a group signature scheme. They used group signatures as the client digital certificate. However, the means for sending such certificates over IP networks were outside the scope of the study. Lee et al. [10] proposed an anonymous subscription service, called Anon-pass. Their construction methodology is similar to group signatures, wherein a user proves the possession of signatures using zero-knowledge proofs. Though Anon-pass does not consider end-to-end secure (encrypted) communication, our protocol does.

Gilad and Herzberg [11] considered the distribution of public keys using an anonymous service. They consider two peers, a querier and a responder. The querier specifies a random ephemeral public key that is not certified by the CA, and sends a query that contains this public key to a responder via an anonymous service such as Tor. The responder replies with a message encrypted by the (anonymous) querier ephemeral public key. However, a responder cannot verify whether a public key is a valid key or a random value because this scheme provides no certification of the public key; moreover, the responder cannot even detect whether the public key was replaced by an attacker. Furthermore, no anonymous user authentication is considered in the Gilad-Herzberg system. In our protocol, the SP can be convinced that a public key (i.e., a temporary ID) will work because arbitrary values can be public keys in IBE systems. In addition, because a temporary ID is signed by group signature, we can prevent a key replacement attack and achieve anonymous user authentication, simultaneously.

Sankey and Wright [12] considered a privacy-preserving next-generation Internet routing protocol called Dovetail that provides anonymity against an active attacker located at any single point within the network (incl. untrusted Internet SPs). Moreover, they insisted that *We do not protect the packet contents, which reside in higher network layers and are thus out of scope for this paper. Content should be protected end-to-end using a protocol such as IKEv2, which protects sender and receiver identities*'', where IKEv2 means Internet Key Exchange Protocol Version 2 [13]. However as mentioned before, certification of a public key contradicts anonymity, and an attacker can replace a public key if no certification is used. Therefore, it is not clear whether Dovetail with IKEv2 provide both anonymity and a secure channel, and we insist that these must be considered simultaneously.

Proxy re-encryption (PRE) (see [14]) is another candidate for building secure channels without conflicting anonymity. In our context, users first compute re-encryption keys using their secret key and the SP public key, and the SP computes ciphertext only using its public key. Then, the proxy can re-encrypt ciphertext. However, the proxy needs to manage all re-encryption keys, and therefore, it is difficult to assume that no private information is infringed even if the proxy is corrupted after communication. Moreover, there is a possibility that other users might decrypt unexpected ciphertext because the proxy manages many re-encryption keys (from the SP to each user). Generating re-encryption keys that can be used in an unexpected manner is undesirable, even if the proxy is modeled as an honest-but-curious entity and always follows protocol. In our protocol no unexpected user (including the proxy) can decrypt ciphertexts because a unique temporary ID is assigned to each user and each session. We note that the key escrow problem occurs as an outcome of IBE, where the key generation center (KGC) can decrypt all ciphertexts. However, KGC is modeled as a trusted third party, whereas it is difficult to fully trust all proxies involved in the systems.

D. DIFFERENCES FROM PROCEEDINGS VERSION

In the proceedings version [15], we provided our experimental results after running the proposed protocol using Simpleproxy. In this paper, we extend the content by adding our experimental results after running the proposed protocol using the Tor network, and confirm that the costs of the cryptographic operations of our protocol are not dominant in this case. More details on the protocol and implementation, as well as a discussion, are elaborated as well; we discuss the revocation functionality of the underlying group signature, and provide the security proofs of our protocol that were omitted in the proceedings version. In addition, we discuss the protocol's compatibility and deployability over the Internet.

E. ORGANIZATION

The remainder of this paper is organized as follows: In Section II, cryptographic tools are introduced; in particular, we introduce the Boneh-Franklin IBE scheme [5] and provide our proposed open-free group signature scheme. In Section III, we introduce the proposed protocol; we begin with its framework, followed by its formal security definition (semantic security, anonymity, and unforgeability) in a cryptographic manner, as well as its construction. We indicate that the protocol uses IBE and group signature as underlying cryptographic tools as well as the proxy module for anonymous communications, and show that the protocol is provably secure within the scope of our definition. In Section IV, we introduce our proof-of-concept implementation that instantiates our protocol. We show that the protocol uses the cryptographic tools and manages Internet communication through a newly defined HTTP method and header. This section also evaluates the efficiency of the proposed protocol when Simpleproxy and Tor are used as the proxy module. In Section V, we discusses compatibility and deployability of the protocol, as well as alternative approaches to the secure and anonymous communication protocol.

II. CRYPTOGRAPHIC TOOLS

A. IDENTITY-BASED ENCRYPTION

In this section, we give the definition of IBE as follows. An IBE scheme $\mathcal{IBE}$ consists of four algorithms: i.e., (IBE.Setup, Extract, IBE.Enc, IBE.Dec). Let $\mathcal{ID}$ and $\mathcal{M}$ be an identity space and message space, respectively.

Definition 1 (Syntax of IBE [5]):

- IBE.Setup*: This algorithm takes as input the security parameter λ, and outputs a public key params and a master secret key msk.*
- Extract*: This algorithm takes as input params, msk, and an identity $ID \in \mathcal{ID}$, and outputs a decryption key dk_{ID}. This algorithm is supposed to be run by KGC.*
- IBE.Enc*: This algorithm takes as input params, ID, and a message $M \in \mathcal{M}$, and outputs a ciphertext C_{IBE}.*
- IBE.Dec*: This algorithm takes as input params, C_{IBE}, and dk_{ID}, and outputs M.*

We require the following correctness property: for all $(params, msk) \leftarrow$ IBE.Setup (1^{λ}), all ID and all M,

$\Pr[\mathsf{IBE.Dec}(params, \mathsf{IBE.Enc}(params, ID, M), \mathsf{Extract}(params, msk, ID)) = M] = 1$ holds.

We require semantic security (i.e., indistinguishability of ciphertexts under an adaptive chosen-identity and chosen-plaintext attack (IND-ID-CPA)) which is defined as follows:

Definition 2 (IND-ID-CPA):

1) *The challenger $\mathcal{C}$ runs $(params, msk) \leftarrow \mathsf{IBE.Setup}(1^\lambda)$, and gives params to an adversary $\mathcal{A}$.*
2) *$\mathcal{A}$ is allowed to issue extract queries ID. $\mathcal{C}$ runs $dk_{ID} \leftarrow \mathsf{Extract}(params, msk, ID)$, and returns dk_{ID} to $\mathcal{A}$.*
3) *$\mathcal{A}$ sends two messages M_0 and M_1, and the challenge identity ID^*, where no extract query for ID^* has been issued. $\mathcal{C}$ flips a coin $b \in \{0, 1\}$, computes $C^*_{IBE} \leftarrow \mathsf{IBE.Enc}(params, ID^*, M_b)$, and returns C^*_{IBE} to $\mathcal{A}$.*
4) *$\mathcal{A}$ is allowed to issue extract queries $ID \neq ID^*$. $\mathcal{C}$ runs $dk_{ID} \leftarrow \mathsf{Extract}(params, msk, ID)$, and returns dk_{ID} to $\mathcal{A}$.*
5) *Finally, $\mathcal{A}$ outputs a bit b'.*

We say that an IBE scheme is IND-ID-CPA secure if $|\Pr[b = b'] - 1/2|$ is negligible against the security parameter λ.

Next, we introduce the Boneh-Franklin IBE scheme [5] as follows. This scheme is IND-ID-CPA secure under the bilinear Diffie-Hellman assumption in the random oracle model. Let $\mathcal{ID}$ and $\mathcal{M}$ be an identity space and message space, respectively.

Proposition 1 (Boneh-Franklin IBE [5]):

- $\mathsf{IBE.Setup}$*: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be a bilinear group with prime order p, where $\langle g_1 \rangle = \mathbb{G}_1$, $\langle g_2 \rangle = \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map.*[2] *Choose $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and compute $P_{pub} = g_1^\alpha$. Output $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, P_{pub}, H_1, H_2)$ and $msk = \alpha$, where $H_1 : \mathcal{ID} \rightarrow \mathbb{G}_2$ and $H_2 : \mathbb{G}_T \rightarrow \mathcal{M}$ are hash functions modeled as random oracles.*
- $\mathsf{Extract}$*: For $ID \in \mathcal{ID}$, compute $H_1(ID)^\alpha \in \mathbb{G}_2$ and output a decryption key $dk_{ID} = H_1(ID)^\alpha$.*
- $\mathsf{IBE.Enc}$*: For $M \in \mathcal{M}$, choose $r \xleftarrow{\$} \mathbb{Z}_p$, compute $C_1 = g_1^r$ and $C_2 = M \oplus H_2(e(P_{pub}, H_1(ID))^r)$, and output $C_{IBE} = (C_1, C_2)$.*
- $\mathsf{IBE.Dec}$*: Output $M = C_2 \oplus H_2(e(C_1, dk_{ID}))$. Note that $e(P_{pub}, H_1(ID))^r = e(g_1^\alpha, H_1(ID))^r = e(g_1^r, H_1(ID)^\alpha) = e(C_1, dk_{ID})$ holds.*

B. GROUP SIGNATURE WITH OPEN-FREE VARIANT

The proposed secure and anonymous communication protocol uses a group signature scheme as its fundamental component. The conventional group signature realizes the open functionality, where an authority called opener can identify who the actual signer is. Since the Proxy module manages source IP address in our protocol, we can regard the Proxy as an opener if the open functionality is realized. Though arbitrary group signature schemes could be used (i.e., by ignoring open functionality), it is beneficial to remove unnecessary functionality and improve performance efficiency. In this section, we newly give definitions of group signature with its open-free variant which we call open-free group signature.[3]

First, we give the syntax of group signature with its open-free variant. An open-free group signature scheme $\mathcal{GS}$ consists of four algorithms: ($\mathsf{GS.Setup}$, Join, Sign, Verify) as follows:

Definition 3 (Syntax of Open-Free Group Signature):

- $\mathsf{GS.Setup}$*: This algorithm takes as input the security parameter λ, and outputs a group public key gpk and an issuer key ik. This algorithm is supposed to be run by GM.*
- $\mathsf{GS.Join}$*: This algorithm takes as input gpk and ik (from GM), and a user is obtained a signing key sk.*
- Sign*: This algorithm takes as input gpk, a signing key sk, and a message M, and outputs a group signature σ.*
- Verify*: This algorithm takes as input gpk, σ, and M, and outputs 1 if σ is a valid signature on M, and 0 otherwise.*

Note that we do not have to assume that the user has a secret key as a input of the $\mathsf{GS.Join}$ *algorithm due to the open-free property.*

We require the following correctness property: for all $(gpk, ik) \leftarrow \mathsf{GS.Setup}(1^\lambda)$ and $sk \leftarrow \mathsf{GS.Join}(gpk, ik)$, $\Pr[\mathsf{Verify}(gpk, \mathsf{Sign}(gpk, sk, M), M) = 1] = 1$ holds.

Next, we redefine the security definitions of the Furukawa-Imai group signature scheme, i.e., anonymity, traceability, and non-frameability, to match the open-free variant. Anonymity guarantees that no adversary $\mathcal{A}$ can distinguish whether two signers of group signatures are the same or not, even if $\mathcal{A}$ has the corresponding signing keys. Usually, there are two kind of anonymity, CPA-anonymity and CCA-anonymity. In CCA-anonymity, $\mathcal{A}$ is allowed to issue open queries, where $\mathcal{A}$ sends (σ, M), and is given the result of the Open algorithm. Meanwhile, we do not have to consider these differences due to the open-free property.

Definition 4 (Anonymity):

1) *An adversary $\mathcal{A}$ with the security parameter λ sends gpk, sk_0, sk_1, and M to the challenger $\mathcal{C}$.*
2) *$\mathcal{C}$ chooses $b \xleftarrow{\$} \{0, 1\}$, computes $\sigma^* \leftarrow \mathsf{Sign}(gpk, sk_b, M)$, and sends σ^* to $\mathcal{A}$.*
3) *$\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.*

An open-free group signature $\mathcal{GS}$ is said to have anonymity if $\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{GS},\mathcal{A}}(\lambda) := |\Pr[b = b'] - 1/2|$ is negligible in λ.

Next, we redefine traceability. In usual definition, traceability guarantees that no adversary $\mathcal{A}$ can produce

[2] We require bilinearity: for all $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$, and non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the identity element in $\mathbb{G}_T$.

[3] This new primitive is a kind of dynamic group signature, where a new member can join the system even after the setup phase. We note that, additional two algorithms, Open and Judge, are usually contained in dynamic group signatures (see [16]). The Open algorithm identifies the actual signer by using the GM's secret key. The Judge algorithm checks a proof output by the Open algorithm, whether the Open algorithm is correctly executed or not. Obviously, the Judge algorithm is meaningless in the open-free variant.

a valid-but-untraceable group signature, that is, the Open algorithm cannot identify the corresponding signer though the Verify algorithm outputs 1. However, in the open-free variant, this definition is meaningless. So, we define unforgeability here instead of traceability, where no adversary $\mathcal{A}$ can produce a valid group signature without knowing a signing key.

Definition 5 (Unforgeability):

1) *The challenger $\mathcal{C}$ runs $(gpk, ik) \leftarrow \mathsf{GS.Setup}(1^\lambda)$, and gives gpk to an adversary $\mathcal{A}$.*
2) *$\mathcal{A}$ is allowed to issue the signing query (i, M). If a user U_i has not been joined to the system, then $\mathcal{C}$ runs the* GS.Join *algorithm, computes sk_i, and returns $\sigma \leftarrow \mathsf{Sign}(gpk, sk_i, M)$ to $\mathcal{A}$. If U_i has been joined to the system, then $\mathcal{C}$ returns $\sigma \leftarrow \mathsf{Sign}(gpk, sk_i, M)$ to $\mathcal{A}$. Moreover, $\mathcal{C}$ appends (σ, M) into the list $\mathcal{S}$.*
3) *Finally, $\mathcal{A}$ outputs (σ^*, M^*). We say that $\mathcal{A}$ wins if $\mathsf{Verify}(gpk, \sigma^*, M^*) = 1$ holds and $(\sigma^*, M^*) \notin \mathcal{S}$.*

An open-free group signature $\mathcal{GS}$ is said to have unforgeability if $\mathsf{Adv}^{\mathsf{un}}_{\mathcal{GS},\mathcal{A}}(\lambda) := \Pr[\mathcal{A}\ wins]$ is negligible in λ.

Finally, we revisit non-frameability. Non-frameability guarantees that no adversary $\mathcal{A}$ can produce a valid group signature whose open result is an honest (i.e., uncorrupted by $\mathcal{A}$) user (say U). Obviously, this definition is meaningless in the open-free variant, and therefore we do not consider non-frameability.

We remark that in order to achieve non-frameability in conventional group signature schemes, a user chooses a secret key *usk*, and is obtained its signing key *sk* by executing the GS.Join algorithm with GM. What is critical, GM cannot know *usk* itself (but can convince that the user knows *usk* by using zero-knowledge proofs). In other words, we can remove a secret key *usk* from the syntax of group signature unless non-frameability is required. This is the reason why we do not require any secret key of users as input of the GS.Join algorithm, and the GS.Join algorithm can be a non-interactive algorithm.

1) BUILDING OPEN-FREE GROUP SIGNATURE

Our group signature scheme modifies the Furukawa-Imai group signature [4]. In the Furukawa-Imai scheme, a user certificate issued by the GM is a short signature [17]. The user proves the possession of the certificate by non-interactive zero-knowledge (NIZK) proofs which are constructed via the Fiat-Shamir conversion [18]. For implementing the Open algorithm, an ElGamal-type double encryption is used over a decisional Diffie-Hellman (DDH)-hard group (in addition to bilinear groups). In our open-free scheme, the DDH-hard group can be removed. Other part is the same as that of the original Furukawa-Imai group signature scheme.

Note that, a simple construction, where for one signature verification/signing key pair (VK, SK), each group member shares SK, can also be seen as an open-free group signature scheme. However, this simple construction never realizes the revocation functionality [19]. Here, we newly construct an open-free group signature scheme, and discuss the revocation functionality of our open-free Furukawa-Imai group signature scheme later.

Construction 1 (Proposed Open-Free Group Signature):

- GS.Setup*: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be a bilinear group with prime order p, where $\langle g_1 \rangle = \mathbb{G}_1$, $\langle g_2 \rangle = \mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. Choose $\gamma \xleftarrow{\$} \mathbb{Z}_p$, and $h \xleftarrow{\$} \mathbb{G}_1$, and compute $W = g_2^\gamma$. Output $gpk = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h, W, e(g_1, g_2), e(g_1, W), e(h, W), e(h, g_2), H_3)$ and $ik = \gamma$, where H_3: $\{0,1\}^* \to \mathbb{Z}_p$ is a hash function modeled as a random oracle.*
- GS.Join*: For a user U_i, choose $x_i \xleftarrow{\$} \mathbb{Z}_p \setminus \{-\gamma\}$ and $y_i \xleftarrow{\$} \mathbb{Z}_p$, compute $A_i = (g_1 h^{-y_i})^{\frac{1}{\gamma + x_i}}$, and output $sk_i = (x_i, y_i, A_i)$.*
- Sign*: Let $sk = (x, y, A)$. Choose $\beta \xleftarrow{\$} \mathbb{Z}_p$, set $\delta = \beta x - y$, and compute $T = Ah^\beta$. Choose $r_x, r_\delta, r_\beta \xleftarrow{\$} \mathbb{Z}_p$, and compute $R = e(h, g_2)^{r_\delta} e(h, W)^{r_\beta} / e(T, g_2)^{r_x}$, $c = H_3(gpk, T, R, M)$, $s_x = r_x + cx$, $s_\delta = r_\delta + c\delta$, and $s_\beta = r_\beta + c\beta$, and output $\sigma = (T, c, s_x, s_\delta, s_\beta)$.*
- Verify*: Compute*

$$R' = \frac{e(h, g_2)^{s_\delta} e(h, W)^{s_\beta}}{e(T, g_2)^{s_x}} \left(\frac{e(T, W)}{e(g_1, g_2)}\right)^{-c}$$

and output 1 if $c = H_3(gpk, T, R', M)$ holds, and 0 otherwise.

Compared to the original Furukawa-Imai scheme, we can reduce three DDH-hard group elements and three $\mathbb{Z}_p$ elements. Accordingly, we can reduce the size of signature by 50% compared to the original Furukawa-Imai group signature scheme.

Note that $e(A, g_2^x W) = e(g_1, g_2) e(h, g_2)^{-y}$ holds if (x, y, A) is a valid certificate. From this equation,

$$\frac{e(T, W)}{e(g_1, g_2)} = \frac{e(h, g_2)^{\beta x - y} e(h, W)^\beta}{e(T, g_2)^x}$$

holds for $T = Ah^\beta$. Therefore,

$$\begin{aligned} &\frac{e(h, g_2)^{s_\delta} e(h, W)^{s_\beta}}{e(T, g_2)^{s_x}} \\ &\quad = \frac{e(h, g_2)^{r_\delta} e(h, W)^{r_\beta}}{e(T, g_2)^{r_x}} \left(\frac{e(h, g_2)^{\beta x - y} e(h, W)^\beta}{e(T, g_2)^x}\right)^c \\ &\quad = R \left(\frac{e(T, W)}{e(g_1, g_2)}\right)^c \end{aligned}$$

holds.

2) REVOCATION

We can additionally consider the revocation function [19]. Actually, the revocation function is indispensable in practice, e.g., any group-oriented cryptographic primitives need revocation since a legitimate user may leave the system or the secret key may be leaked. However, in the anonymous environment how to realize revocation function is not trivial,

since no verifier can know whether a signer is a revoked member or not.

Currently, the following revocation methodologies have been proposed. Let r be the number of revoked users and N be the number of users.

1) A signer and the GM update a signing key and gpk, respectively, using the revocation list RL. Then, the signing key update cost is $O(r)$, see [4], whereas the verification cost is $O(1)$.
2) Though the verification cost is $O(r)$, the signing cost is $O(1)$ and no signer is involved in the revocation procedure (verifier-local revocation (VLR)), see [20], [21].
3) Though signing/verification costs are constant, the size of gpk is $O(\sqrt{N})$ [22] or the size of RL is $O(N)$ [23].
4) All costs are asymptotically quite efficient (less than $O(\log N)$), but the real costs are inefficient [19], [24], [25]. For example, a group signature contains almost 100 group elements.

For the open-free group signature scheme, we applied the first revocation methodology as follows. Let a user whose signing key is $sk_j = (A_j, x_j, y_j)$ be revoked. Then, the GM updates $\tilde{g}_1 \leftarrow g_1^{\frac{1}{\gamma+x_j}}$ and $\tilde{h} \leftarrow h^{\frac{1}{\gamma+x_j}}$, and publishes $(\tilde{g}_1, \tilde{h}, (x_j, y_j))$ as the revocation list RL of the current revocation epoch. We denote it $sk_j \in RL$. Then, any user whose signing key is (A_i, x_i, y_i) where $i \neq j$ (and thus $x_i \neq x_j$) can compute $\tilde{A}_i = (\tilde{g}_1 \tilde{h}^{-y_i})^{\frac{1}{\gamma+x_i}}$ for the new $(\tilde{g}_1, \tilde{h})$ such that $\left((A_i/\tilde{g}_1)\tilde{h}^{y_i}\right)^{\frac{1}{x_j-x_i}}$ since

$$(A_i/\tilde{g}_1)\tilde{h}^{y_i} = \left((g_1 h^{-y_i})^{\frac{1}{\gamma+x_i}}/g_1^{\frac{1}{\gamma+x_j}}\right)h^{\frac{y_i}{\gamma+x_j}} = g_1^{\frac{x_j-x_i}{(\gamma+x_i)(\gamma+x_j)}} \cdot h^{\frac{-y_i(x_j-x_i)}{(\gamma+x_i)(\gamma+x_j)}} = (\tilde{g}_1\tilde{h}^{-y_i})^{\frac{x_j-x_i}{\gamma+x_i}}$$

hold. If multiple numbers of users are revoked simultaneously, then this procedure is sequentially run.

C. SECURITY PROOFS OF OUR GROUP SIGNATURE

The remaining part is to show that the proposed open-free group signature scheme is anonymous and unforgeable. The proposed open-free group signature scheme is constructed from an (honest-verifier) zero-knowledge proof of knowledge by using the Fiat-Shamir conversion [18]. First, we explain the original proof of knowledge protocol as follows. A prover computes (T, R), and sends it to a verifier. The verifier sends a challenge value c to the prover. The prover computes (s_x, s_δ, s_β), and sends it to the verifier. The verifier checks whether the verification equation $R = \frac{e(h,g_2)^{s_\delta} e(h,W)^{s_\beta}}{e(T,g_2)^{s_x}} \left(\frac{e(T,W)}{e(g_1,g_2)}\right)^{-c}$ holds or not. Next, we show that this 3-move protocol is zero-knowledge (this immediately leads to anonymity). The simulator chooses $A \xleftarrow{\$} \mathbb{G}$ and $\beta \xleftarrow{\$} \mathbb{Z}_p$, and computes $T = Ag_1^{\beta}$. Note that β is chosen uniformly random. Therefore, T generated from the simulator is drawn from a distribution that is indistinguishable from the distribution output by any particular prover. For $T \in \mathbb{G}$, the simulator chooses $c, s_x, s_\delta, s_\beta \xleftarrow{\$} \mathbb{Z}_p$, and computes $R = \frac{e(h,g_2)^{s_\delta} e(h,W)^{s_\beta}}{e(T,g_2)^{s_x}} \left(\frac{e(T,W)}{e(g_1,g_2)}\right)^{-c}$. Then the transcript $(T, R, c, s_x, s_\delta, s_\beta)$ here is indistinguishable from transcripts of the actual protocol.

Next, we show that the protocol is a proof of knowledge. That is, we show there exists an extractor that can extract a SDH pair from $(T, R, c, s_x, s_\delta, s_\beta)$ and $(T, R, c', s'_x, s'_\delta, s'_\beta)$, where $c \neq c'$ and both transcripts satisfy the verification equation. Set $\tilde{x} := \frac{s_x - s'_x}{c-c'}$, $\tilde{y} := \frac{(s_x - s'_x)(s_\beta - s'_\beta) - (s_\delta - s'_\delta)(c-c')}{(c-c')^2}$, and $\tilde{\beta} := \frac{s_\beta - s'_\beta}{c-c'}$. Then, $\frac{e(T,W)}{e(g_1,g_2)} = \frac{e(h,g_2)^{\tilde{\beta}\tilde{x}-\tilde{y}} e(h,W)^{\tilde{\beta}}}{e(T,g_2)^{\tilde{x}}}$ holds. Therefore, for $\tilde{A} = T/h^{\tilde{\beta}}$, $e(\tilde{A}, g^{\tilde{x}}W) = e(g_1, g_2)e(h, g_2)^{-\tilde{y}}$ holds. That is, $(\tilde{x}, \tilde{y}, \tilde{A})$ can be extracted. Briefly, by the above extractor and the Forking Lemma [26], the simulator rewinds the adversary, obtains two forged signatures, and can extract a SDH pair from forged signatures. This immediately leads to unforgeability.

III. SECURE AND ANONYMOUS COMMUNICATION PROTOCOL

This section proposes a secure and anonymous communication protocol. First, the section describes the protocol framework as an overview of the system. Then, the section defines the protocol syntax and provides its construction. We consider a scenario in which an SP is modeled as a server that provides service only to legitimate users. That is, we can assume that the GM has authenticated a user before issuing a signing key, and the SP can determine whether the user that can generate a valid group signature is legitimate.

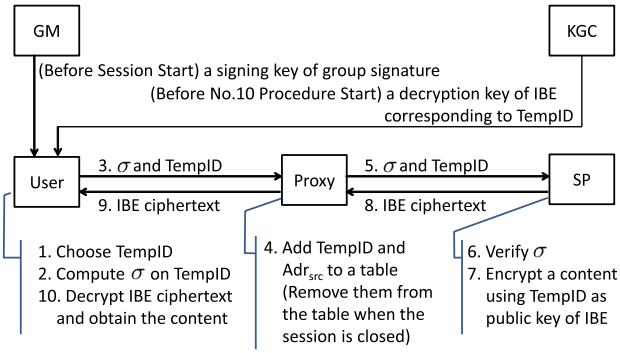


FIGURE 1. Framework of the proposed protocol.

A. FRAMEWORK

In this section, we describe our protocol framework. Figure 1 depicts the framework of the proposed protocol, which has five roles; User, Proxy, SP, GM, and KGC. A **User** wants to communicate with an SP without revealing its identity. Here, we assume that users will never lie about their IP address. The **Proxy** assists communication between the User and SP by relaying packets without revealing the User IP address. We assume that the SP is honest-but-curious. The **SP** provides services to the User, but wants to authenticate it. The SP is not interested in the identity of the User but needs to

confirm whether the User is legitimate. The **GM** manages a group key and issuer key, and issues a signing key to the User to be used for generating an anonymously-authenticated token. We assume that the GM suitably authenticates the User before issuing the signing key. The **KGC** generates a decryption key for the User. We assume that the KGC suitably authenticates the User before issuing the decryption key.

These roles need to collaborate mutually in order to realize the proposed secure anonymous authentication. Their interaction sequence is as follows. (1) A User (whose IP address is $\mathtt{Adr}_{\mathtt{src}}$) chooses a temporary ID $\mathtt{TempID}$, (2) the User computes a group signature σ on $\mathtt{TempID}$, and (3) the User sends $(\sigma, \mathtt{TempID})$ to the Proxy. (4) The Proxy associates $\mathtt{Adr}_{\mathtt{src}}$ with this temporary ID, and (5) forwards $(\sigma, \mathtt{TempID})$ to the SP. (6) The SP can directly authenticate the users by verifying the group signature without compromising anonymous communication. (7) If the user is successfully verified, the SP encrypts content using TempID as the public key of IBE; otherwise, it returns $\perp$. Here, we apply an IBE property to establish a secure channel between the SP and an anonymous user, where arbitrary values can be a public key, and a ciphertext can be independently computed with the generation of the corresponding decryption key.[4] (8) The SP sends this IBE ciphertext to the Proxy, which again (9) forwards it to the corresponding user. (10) Finally, the User decrypts the IBE ciphertext using the corresponding decryption key issued by the KGC. After relaying (mutual) communication, the Proxy immediately deletes the corresponding pair of $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$. Therefore, no private information is infringed even if the Proxy is corrupted after communication. We note that Figure 1 explains one-pass communication. The Proxy can reuse the pair information $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ of a session provided that the session is current, but it removes the information from its registry once the session is closed. In either case, the proxy immediately deletes the corresponding pair $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ after the session. Moreover, we can easily extend one-proxy setting to multi-proxy setting because the Proxy has to do is (1) manage $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$, and (2) forwarding $(\sigma, \mathtt{TempID})$ to the next proxy. Our framework focuses on encrypted communication from the SP to users. Furthermore, the framework can easily be extended to interactive secure communication because SP is not anonymous to users, and thus, each user can simply use the SP public key to build a secure channel.

We state that our framework achieves sending a token anonymously, and that it is not an authentication protocol in the strict sense. However, we can construct an authentication protocol via the classical challenge-and-response methodology: a n SP sends a random nonce to a User (via the Proxy) and the User computes a group signature whose signed message contains the nonce. Therefore, in this paper, we focus mainly on sending a token anonymously and on establishing a secure channel.

[4]This property is used in timed-release encryption [27] context, where an encryptor can control when ciphertexts will be decrypted.

B. SYNTAX AND SECURITY DEFINITIONS

Let $\mathcal{ID}$ and $\mathcal{M}$ be an identity space and message space, respectively, and $\mathtt{Adr}_{\mathtt{src}}$, $\mathtt{Adr}_{\mathtt{proxy}}$, and $\mathtt{Adr}_{\mathtt{dst}}$ stand for IP address of User, Proxy, and SP, respectively. For a set X and an element $x \in X$, $x \xleftarrow{\$} X$ means that x is randomly chosen from X.

Definition 6 (Syntax of the Protocol):

- GM.Setup*: This probabilistic algorithm takes as input the security parameter λ, and outputs a group public key gpk and an issuer key ik.*
- KGC.Setup*: This probabilistic algorithm takes as input the security parameter λ, and outputs a public key params and a master secret key msk.*
- Join*: This probabilistic algorithm takes as input gpk and ik, and outputs a signing key sk.*
- UserKeyGen*: This (possibly) probabilistic algorithm takes as input params, msk, and an (possibly temporary) identity $\mathtt{TempID} \in \mathcal{ID}$, and outputs a decryption key $dk_{\mathtt{TempID}}$.*
- SendRequest*: This probabilistic algorithm takes as input gpk, sk, $\mathtt{TempID}$, a source IP address $\mathtt{Adr}_{\mathtt{src}}$, a destination IP address $\mathtt{Adr}_{\mathtt{dst}}$, and a proxy IP address $\mathtt{Adr}_{\mathtt{proxy}}$, and send a token σ, $\mathtt{TempID}$ and $\mathtt{Adr}_{\mathtt{dst}}$ to the proxy whose IP address is $\mathtt{Adr}_{\mathtt{proxy}}$.*
- RelayRequest*: This deterministic algorithm takes as input $\mathtt{Adr}_{\mathtt{src}}$, $\mathtt{Adr}_{\mathtt{dst}}$, an ID/IP table $\mathtt{Tbl}$, σ, and $\mathtt{TempID}$, and relays a pair $(\sigma, \mathtt{TempID})$ and $\mathtt{Adr}_{\mathtt{proxy}}$ to the destination SP whose IP address is $\mathtt{Adr}_{\mathtt{dst}}$. Moreover, append $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ to $\mathtt{Tbl}$.*
- ValidityCheck*: This deterministic algorithm takes as input gpk, σ, and $\mathtt{TempID}$, and outputs 1 if σ is valid, and 0, otherwise.*
- SendContent*: This probabilistic algorithm takes as input gpk, σ, $\mathtt{TempID}$, a content to be sent $M \in \mathcal{M}$, and $\mathtt{Adr}_{\mathtt{proxy}}$, computes a ciphertext C if the token σ is valid, and sends C to the proxy whose IP address is $\mathtt{Adr}_{\mathtt{proxy}}$. Otherwise, if σ is invalid, then return $\perp$.*
- RelayContent*: This deterministic algorithm takes as input C and $\mathtt{Tbl}$, and relays C to a user whose IP address is $\mathtt{Adr}_{\mathtt{src}}$ contained in $\mathtt{Tbl}$. Moreover, remove $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ from $\mathtt{Tbl}$.*
- GetContent*: This deterministic algorithm takes as input C and $dk_{\mathtt{TempID}}$, and return M.*

Next, we give formal security definitions as follows. First, we define correctness that guarantees σ is valid and a user always can obtain the corresponding content if all values are honestly generated according to the algorithms.

Definition 7 (Correctness): For all $(gpk, ik) \leftarrow$ GM.Setup (1^λ), $(params, msk) \leftarrow$ KGC.Setup (1^λ), $sk \leftarrow$ Join (gpk, ik), $\mathtt{TempID} \in \mathcal{ID}$, $M \in \mathcal{M}$, *and*

$(\mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$,

$$\Pr[M \leftarrow \mathsf{GetContent}\Big(\mathsf{RelayContent}\big(C, \mathtt{Tbl}\big), \mathsf{UserKeyGen}(\mathit{param}, \mathit{msk}, \mathtt{TempID})\Big)] = 1, \text{ and}$$

$$\Pr[1 \leftarrow \mathsf{ValidityCheck}(\mathit{gpk}, \sigma, \mathtt{TempID}) = 1] = 1$$

where $(\sigma, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{dst}}) \leftarrow \mathsf{SendRequest}(\mathit{gpk}, \mathit{sk}, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$, $(\sigma, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{proxy}}) \leftarrow \mathsf{RelayRequest}(\mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Tbl}, \sigma, \mathtt{TempID})$, *and* $C \leftarrow \mathsf{SendContent}(\mathit{gpk}, \sigma, \mathtt{TempID}, M, \mathtt{Adr}_{\mathtt{proxy}})$.

Next, we define anonymity, semantic security, and unforgeability as follows. One session is defined as sequences of algorithm executions from SendRequest to GetContent, where SendRequest → RelayRequest → SendContent → RelayContent → GetContent. Anonymity guarantees that no adversary $\mathcal{A}$ who is allowed to communicate with the proxy (but is not allowed to know $\mathtt{Adr}_{\mathtt{src}}$) can distinguish whether the users of two different sessions are the same or not. In this game, $\mathcal{A}$ is modeled as a malicious SP. Moreover, we care about signing key exposure, where $\mathcal{A}$ can obtain signing keys. In addition to this, we give *msk* to $\mathcal{A}$ in order to guarantee that the KGC ability has nothing to right for identifying the user.[5]

Definition 8 (Anonymity):

1) *The challenger* $\mathcal{C}$ *runs* $(\mathit{gpk}, \mathit{ik}) \leftarrow \mathsf{GM.Setup}(1^\lambda)$ *and* $(\mathit{params}, \mathit{msk}) \leftarrow \mathsf{KGC.Setup}(1^\lambda)$*, and gives two signing keys* $\mathit{sk}_0, \mathit{sk}_1 \leftarrow \mathsf{Join}(\mathit{gpk}, \mathit{ik})$*, and gives* gpk*,* sk_0*,* sk_1*, and* $(\mathit{params}, \mathit{msk})$ *to an adversary* $\mathcal{A}$*. Moreover,* $\mathcal{C}$ *initializes* $\mathtt{Tbl} := \emptyset$.
2) $\mathcal{A}$ *is allowed to issue the* SendRequest *query* $(i, \mathtt{TempID}) \in \{0,1\} \times \mathcal{ID}$*.* $\mathcal{C}$ *runs* $\mathsf{SendRequest}(\mathit{gpk}, \mathit{sk}_b, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$*, and returns* σ *(generated via the* SendRequest *algorithm) to* $\mathcal{A}$.
3) $\mathcal{A}$ *is allowed to issue the* RelayRequest *query* $(\sigma, \mathtt{TempID})$*.* $\mathcal{C}$ *runs* $\mathsf{RelayRequest}(\mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Tbl}, \sigma, \mathtt{TempID})$ *and updates* $\mathtt{Tbl}$.
4) $\mathcal{A}$ *is allowed to issue the* RelayContent *query* C*.* $\mathcal{C}$ *runs* $\mathsf{RelayContent}(C, \mathtt{Tbl})$*, and updates* $\mathtt{Tbl}$.
5) $\mathcal{A}$ *sends* $\mathtt{TempID}^* \in \mathcal{ID}$ *to* $\mathcal{C}$*.* $\mathcal{C}$ *flips a coin* $b \xleftarrow{\$} \{0,1\}$*, and runs* $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{dst}}) \leftarrow \mathsf{SendRequest}(\mathit{gpk}, \mathit{sk}_b, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$ *and* $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{proxy}}) \leftarrow \mathsf{RelayRequest}(\mathit{Adr}_{\mathit{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Tbl}, \sigma^*, \mathtt{TempID}^*)$*.* $\mathcal{A}$ *returns an arbitrary* C *to* $\mathcal{C}$*.* $\mathcal{C}$ *runs* $C \leftarrow \mathsf{RelayContent}(C, \mathtt{Tbl})$*. We note that* $\mathcal{A}$ *can know the transcript of these algorithms executed by* $\mathcal{C}$*:* $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{dst}})$*,* $(\sigma^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{proxy}})$*, and* C*.* $\mathcal{A}$ *outputs* $b' \in \{0,1\}$.

The protocol is said to have anonymity if $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{pro},\mathcal{A}}(\lambda) := |\Pr[b = b'] - 1/2|$ *is negligible in* λ.

[5]We remark that we exclude the trivial case that KGC is offered to generate a decryption key of $\mathtt{TempID}$ from a user whose IP address is $\mathtt{Adr}_{\mathtt{src}}$, and observes that the transcript containing $\mathtt{TempID}$.

Next, we define semantic security which guarantees that no information of content M is revealed from the transcripts of algorithms. In this game, an adversary $\mathcal{A}$ is modeled as a malicious proxy. Moreover, $\mathcal{A}$ is allowed to obtain *ik* in order to guarantee that no information of M is revealed even from the GM's viewpoint.

Definition 9 (Semantic Security):

1) *The challenger* $\mathcal{C}$ *runs* $(\mathit{gpk}, \mathit{ik}) \leftarrow \mathsf{GM.Setup}(1^\lambda)$ *and* $(\mathit{params}, \mathit{msk}) \leftarrow \mathsf{KGC.Setup}(1^\lambda)$*, and gives gpk, ik, and params to an adversary* $\mathcal{A}$.
2) $\mathcal{A}$ *is allowed to issue the* UserKeyGen *query* $\mathtt{TempID} \in \mathcal{ID}$*.* $\mathcal{C}$ *runs* $\mathsf{UserKeyGen}(\mathit{params}, \mathit{msk}, \mathtt{TempID})$ *and returns* $dk_{\mathtt{TempID}}$.
3) $\mathcal{A}$ *sends* $\mathtt{TempID}^* \in \mathcal{ID}, M_0^*, M_1^* \in \mathcal{M}$ *and* sk^* *to* $\mathcal{C}$ *as his choice, where* $\mathtt{TempID}^*$ *has not been sent as a* UserKeyGen *query.* $\mathcal{C}$ *flips a coin* $b \xleftarrow{\$} \{0,1\}$*, runs* $\mathsf{SendRequest}(\mathit{gpk}, \mathit{sk}^*, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$ *and* $C^* \leftarrow \mathsf{SendContent}(\mathit{gpk}, \sigma, \mathtt{TempID}, M_b^*, \mathtt{Adr}_{\mathtt{proxy}})$*, and sends* $(\sigma, \mathtt{TempID}^*, \mathtt{Adr}_{\mathtt{dst}})$ *and* C^* *to* $\mathcal{A}$.
4) $\mathcal{A}$ *is allowed to issue the* UserKeyGen *query* $\mathtt{TempID} \in \mathcal{ID}$ *where* $\mathtt{TempID} \neq \mathtt{TempID}^*$*.* $\mathcal{C}$ *runs* $\mathsf{UserKeyGen}(\mathit{params}, \mathit{msk}, \mathtt{TempID})$ *and returns* $dk_{\mathtt{TempID}}$.
5) *Finally,* $\mathcal{A}$ *outputs* $b' \in \{0,1\}$.

The protocol is said to have semantic security if $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pro},\mathcal{A}}(\lambda) := |\Pr[b = b'] - 1/2|$ *is negligible in* λ.

Finally, we define unforgeability which guarantees that no adversary $\mathcal{A}$ who does not have a signing key will be accepted by the ValidityCheck algorithm. In this game, $\mathcal{A}$ is modeled as a malicious user. Moreover, $\mathcal{A}$ is allowed to obtain *msk* in order to guarantee that nobody can be accepted by SP even by KGC.

Definition 10 (Unforgeability):

1) *The challenger* $\mathcal{C}$ *runs* $(\mathit{gpk}, \mathit{ik}) \leftarrow \mathsf{GM.Setup}(1^\lambda)$ *and* $(\mathit{params}, \mathit{msk}) \leftarrow \mathsf{KGC.Setup}(1^\lambda)$*, and gives gpk and* $(\mathit{params}, \mathit{msk})$ *to an adversary* $\mathcal{A}$*. Moreover,* $\mathcal{C}$ *initializes* $\mathcal{S} = \emptyset$.
2) $\mathcal{A}$ *is allowed to issue the* SendRequest *query* $(i, \mathtt{TempID})$*. If* sk_i *has not been generated, then* $\mathcal{C}$ *runs* $\mathit{sk}_i \leftarrow \mathsf{Join}(\mathit{gpk}, \mathit{ik})$ *and preserves* sk_i*.* $\mathcal{C}$ *runs* $\mathsf{SendRequest}(\mathit{gpk}, \mathit{sk}_i, \mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}}, \mathtt{Adr}_{\mathtt{dst}}, \mathtt{Adr}_{\mathtt{proxy}})$*, and sends* σ *to* $\mathcal{A}$*. Moreover,* $\mathcal{C}$ *appends* $(\sigma, \mathtt{TempID})$ *to* $\mathcal{S}$.
3) *Finally,* $\mathcal{A}$ *outputs* $(\sigma^*, \mathtt{TempID}^*)$*. We say that* $\mathcal{A}$ *wins if* $(\sigma^*, \mathtt{TempID}^*) \notin \mathcal{S}$ *and* $\mathsf{ValidityCheck}(\mathit{gpk}, \sigma^*, \mathtt{TempID}^*) = 1$.

The protocol is said to have unforgeability if $\mathsf{Adv}^{\mathsf{uf}}_{\mathsf{pro},\mathcal{A}}(\lambda) := \Pr[\mathcal{A}\ \mathit{wins}]$ *is negligible in* λ.

We say that a protocol is called secure anonymous authentication protocol if the protocol is correct and has anonymity, semantic security, and unforgeability. We remark that we need to assume that (1) the GM and KGC need to be fully trusted, (2) the proxy does not collude with the SP, and (3) users never share their signing key with other people if ALL

security notions (anonymity, semantic security, and unforgeability) are required to be hold.

C. PROTOCOL CONSTRUCTION

In this section, we give our proposed construction. Let (IBE.Setup, Extract, IBE.Enc, IBE.Dec) be an IBE scheme, and (GS.Setup, Join, Sign, Verify) be an open-free group signature scheme. In this construction, a signed message of the underlying group signature is $\texttt{TempID}$ which is also regarded as a public key of the underlying IBE.

Corollary 1 (Proposed Protocol):

- GM.Setup*: Run* $(gpk, ik) \leftarrow$ GS.Setup (1^λ)*, and output* (gpk, ik).
- KGC.Setup*: Run* $(params, msk) \leftarrow$ IBE.Setup (1^λ)*, and output* $(params, msk)$.
- Join*: Run* $sk \leftarrow$ GS.Join(gpk, ik)*, and output* sk.
- UserKeyGen*: Run* $dk_{\texttt{TempID}} \leftarrow$ Extract $(params, msk, \texttt{TempID})$*, and output* $dk_{\texttt{TempID}}$.
- SendRequest*: Choose* $\texttt{TempID} \xleftarrow{\$} \mathcal{ID}$*. Run* $\sigma \leftarrow$ Sign $(gpk, sk, \texttt{TempID})$*, and send* $(\sigma, \texttt{TempID}, \texttt{Adr}_{\texttt{dst}})$ *to the proxy whose IP address is* $\texttt{Adr}_{\texttt{proxy}}$.
- RelayRequest*: Append* $(\texttt{TempID}, \texttt{Adr}_{\texttt{src}})$ *to* $\texttt{Tbl}$*, and relays a pair* $(\sigma, \texttt{TempID})$ *and* $\texttt{Adr}_{\texttt{proxy}}$ *to the destination SP whose IP address is* $\texttt{Adr}_{\texttt{dst}}$.
- ValidityCheck*: Output* 1 *if* Verify $(gpk, \sigma, \texttt{TempID}) = 1$*, and* 0*, otherwise.*
- SendContent*: Output* $\perp$ *if* ValidityCheck $(gpk, \sigma, \texttt{TempID}) = 0$*. Otherwise, run* $C_{IBE} \leftarrow$ IBE.Enc $(params, \texttt{TempID}, M)$*, and send* C_{IBE} *to the proxy whose IP address is* $\texttt{Adr}_{\texttt{proxy}}$.
- RelayContent*: Relay* C_{IBE} *to a user whose IP address is* $\texttt{Adr}_{\texttt{src}}$ *contained in* $\texttt{Tbl}$*. Moreover, remove* $(\texttt{TempID}, \texttt{Adr}_{\texttt{src}})$ *from* $\texttt{Tbl}$.
- GetContent*: Output the result of* IBE.Dec $(params, C_{IBE}, dk_{\texttt{TempID}})$.

We note that our construction only considers one-proxy setting, and therefore no anonymity is guaranteed from the viewpoint of the Proxy. This situation does not contradict our definition of anonymity (Def. 8). We can simply extend this protocol to a multi-proxy setting, where each Proxy relays $(\sigma, \texttt{TempID})$ or C_{IBE} between the previous Proxy and the next Proxy. Then, anonymity is guaranteed even from the Proxies' point of view unless all Proxies collude with each other.

D. SECURITY PROOFS OF PROPOSED PROTOCOL

Theorem 1: Our protocol is anonymous if the underlying group signature scheme is anonymous.

Proof: Let $\mathcal{A}$ be an adversary who can break anonymity of our protocol. Then, we can construct an algorithm $\mathcal{B}$ that breaks anonymity of the underlying group signature scheme as follow. Let $\mathcal{C}$ be the challenger of the underlying group signature. $\mathcal{B}$ generates gpk, sk_0, and sk_1, and generates all IBE-related values. Then $\mathcal{B}$ gives $(gpk, sk_0, sk_1, params, msk)$ to $\mathcal{A}$. In the challenge phase, $\mathcal{B}$ gets $\texttt{TempID}^*$ from $\mathcal{A}$, forwards it to $\mathcal{C}$, and gets σ^* from $\mathcal{C}$. $\mathcal{B}$ uses σ^* as the output of the SendRequest algorithm, and similarly simulates other algorithms. $\mathcal{A}$ outputs b' and $\mathcal{B}$ also outputs b' as the guessing bit. Then, $\mathcal{B}$ can break anonymity of the group signature with the same advantage of $\mathcal{A}$. This contradicts that the underlying group signature is anonymous. □

Theorem 2: Our protocol is semantic secure if the underlying IBE scheme is IND-ID-CPA secure.

Proof: Let $\mathcal{A}$ be an adversary who can break semantic security of our protocol. Then, we can construct an algorithm $\mathcal{B}$ that breaks IND-ID-CPA security of the underlying IBE scheme as follows. Let $\mathcal{C}$ be the challenger of the underlying IBE. $\mathcal{C}$ runs $(params, msk) \leftarrow$ IBE.Setup(1^λ), and sends $params$ to $\mathcal{B}$. $\mathcal{B}$ runs $(gpk, ik) \leftarrow$ GS.Setup(1^λ), and sends (gpk, ik) and $params$ to $\mathcal{A}$. If $\mathcal{A}$ issues a key extraction query $\texttt{TempID}$, then $\mathcal{B}$ forwards $\texttt{TempID}$ to $\mathcal{C}$, obtains $dk_{\texttt{TempID}}$, and returns $dk_{\texttt{TempID}}$ to $\mathcal{A}$. $\mathcal{A}$ sends $\texttt{TempID}^*$, M_0, M_1, and sk^* to $\mathcal{B}$. $\mathcal{B}$ forwards $\texttt{TempID}^*$, M_0, and M_1 to $\mathcal{C}$. $\mathcal{C}$ computes C^*_{IBE}, and sends it to $\mathcal{B}$. $\mathcal{B}$ computes σ on $\texttt{TempID}^*$ using sk^*, and sends $(\sigma, \texttt{TempID}^*, C^*_{IBE})$ to $\mathcal{A}$. If $\mathcal{A}$ issues a key extraction query $\texttt{TempID} \neq \texttt{TempID}^*$, then $\mathcal{B}$ forwards $\texttt{TempID}$ to $\mathcal{C}$, obtains $dk_{\texttt{TempID}}$, and returns $dk_{\texttt{TempID}}$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $b' \in \{0, 1\}$, and $\mathcal{B}$ outputs b' and breaks IND-ID-CPA security of the underlying IBE scheme. □

Theorem 3: Our protocol is unforgeable if the underlying group signature scheme is unforgeable.

Proof: Let $\mathcal{A}$ be an adversary who can break unforgeability of our protocol. Then, we can construct an algorithm $\mathcal{B}$ that breaks unforgeability of the underlying group signature scheme as follows. Let $\mathcal{C}$ be the challenger of the underlying group signature. $\mathcal{C}$ runs $(gpk, ik) \leftarrow$ GM.Setup(1^λ) and gives gpk to $\mathcal{B}$. $\mathcal{B}$ runs $(params, msk) \leftarrow$ KGC.Setup(1^λ), and gives gpk and $(params, msk)$ to $\mathcal{A}$. If $\mathcal{A}$ sends $(i, \texttt{TempID})$ to $\mathcal{B}$, then $\mathcal{B}$ forwards $(i, \texttt{TempID})$ to $\mathcal{C}$ as a signing query. $\mathcal{C}$ runs SendRequest$(gpk, sk_i, \texttt{TempID}, \texttt{Adr}_{\texttt{src}}, \texttt{Adr}_{\texttt{dst}}, \texttt{Adr}_{\texttt{proxy}})$, and sends σ to $\mathcal{B}$. $\mathcal{B}$ sends σ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $(\sigma^*, \texttt{TempID}^*)$. $\mathcal{B}$ outputs $(\sigma^*, \texttt{TempID}^*)$ and breaks unforgeability of the group signature scheme. □

In summary, our protocol is secure (i.e., semantic secure, anonymous, and unforgeable) if the underlying IBE and group signature are secure.

IV. PROOF-OF-CONCEPT IMPLEMENTATION

This section introduces a prototype that implements the proposed protocol and evaluates its performance to demonstrate the feasibility and practicality of the protocol.

A. COMMUNICATION SEQUENCES

Three types of communication sequences are implemented: User-GM, User-KGC, and User-Proxy-SP, and each of the sequences runs the modules defined in Section III-C.

The **User-GM** sequence begins with the Join module, which communicates with the GM. The GM then computes the signing key sk, and returns it to the User. This sequence needs to be run before the User-Proxy-SP sequence starts. For simplicity, the proof-of-concept implementation runs the

User-GM sequence manually and obtains σ prior to the start of the User-Proxy-SP sequence.

The **User-KGC** sequence begins with the UserKeyGen module, which communicates with the KGC. The User sends $\mathtt{TempID}$ to the KGC, and the KGC then computes the decryption key $dk_{\mathtt{TempID}}$, and returns it to the User. This User-KGC sequence and the User-Proxy-SP sequences are run in parallel, though the User-KGC procedure needs to be completed before the GetContent module of the User-Proxy-SP procedure is run. For simplicity, the proof-of-concept implementation runs this sequence manually and obtains $dk_{\mathtt{TempID}}$ prior to the start of the User-Proxy-SP sequence.

The **User-Proxy-SP** sequence begins with the SendRequest module that sends a group signature and TempID. Upon receiving them, the Proxy registers each pair $(\mathtt{TempID}, \mathtt{Adr}_{\mathtt{src}})$ in a table and runs the RelayRequest module that forwards the pair to the SP. The SP then runs the ValidityCheck module as well as the SendContent module that returns an IBE ciphertext to the Proxy, which forwards that to the User. The User then runs the GetContent module that decrypts the IBE ciphertext using the corresponding $dk_{\mathtt{TempID}}$.

We use the HTTP protocol for communication between roles because the modules we use for Proxy, i.e., Simpleproxy and Tor routers, support the HTTP protocol. In order to integrate the proposed protocol with the HTTP protocol, we define an additional method for the HTTP protocol, called "A-GET", and a new header, called A-Authorization.[6] When we use A-GET method, we are required to use the A-Authorization header that embeds the σ and TempID, as described in Figure 2. Note that we have used "*****" for the separator.

```
A-Get [path name] HTTP/1.0 \r\n
Host: [host name]: [port number] \r\n
A-Authorization: σ [separator] TempID \r\n
```

FIGURE 2. A-GET method and A-authorization header

B. MODULE SOFTWARE IMPLEMENTATION

The roles are implemented as separate modules. The **User** role is implemented using C language (GCC version 4.2.1). For proof-of-concept implementation, we have prepared a simple client software that supports our protocol. The software uses the aforementioned A-GET method and A-Authorization header so that it can convey group signature and TempID. The software has a command line interface, through which we can send a request to the designated IP address. After sending the request, this software awaits a response from the SP and displays the response upon receiving it. Figure 3 describes the HTTP communication log of the prototype.

[6]We added anonymity to the existing protocol, thus we have put the prefix of "A- to these newly defined method and header.

```
A-GET /index.html HTTP/1.0
Host: 192.168.35.160:12345
A-Authorization: *****
*****BCMCeW/ [omitted]
*****pKSkpKSk [omitted]
*****
```

(a)

```
HTTP/1.0 200 OK
Content-type: text/html
It works!!
```

(b)

FIGURE 3. Prototype's HTTP communication. (a) HTTP request. (b) HTTP response.

The **SP** role is implemented using C language (GCC version 4.2.1). Because this is a proof-of-concept implementation, its functionality is minimal. This software awaits requests from the User. Upon receiving a request from the User, the software replies with a simple HTML document to the User if the request has the Get method, whereas the software runs decryption and encryption using TempID and anonymous authentication using σ before replying to the User if the request has the A-GET method. The software simply replies with a simple HTTP document to requesters upon receiving GET or A-GET messages from them.

The **Proxy** role is implemented using the Simpleproxy [28] and Tor [3] software. A Simpleproxy generates a child process upon receiving a packet from the User. The process opens a new connection with the destination, i.e. SP, and forwards the packet to the SP. This process maintains separate sessions with the User and with the SP. In this way, any information lower than the session layer is concealed; the IP address information and the port information are hidden. When the session is closed, the process is also closed; one child process is responsible for only one session. In our protocol, we prepared a new software, called A-Proxy, which is generated by modifying the Simpleproxy. When the child process is generated, A-Proxy extracts the TempID value from the packet sent from the User and stores it inside its repository along with the source and destination IP address. When the connection is closed, the process deletes the stored information before closing the process. The User-Proxy-SP sequence with the A-Proxy is summarized in Figure 4.

In the case of using Tor as our Proxy, we need to consider the way in which the User connects to a Tor router because the router requires communication using the Socks protocol. Because our User module does not support the Socks protocol, we use the A-Proxy and torsocks modules provided by the Tor project [3]. Then, the User connects to the torsocks module of a Tor router via our A-Proxy. In this way, the Simpleproxy can relay the request from the User to a Tor Router and can manage the A-GET request as mentioned above. The User-Proxy-SP sequence with the Tor network is summarized in Figure 5.

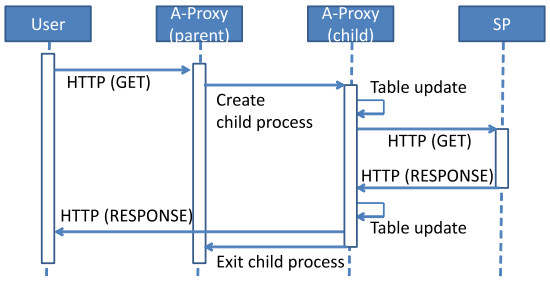


FIGURE 4. The user-proxy-SP sequence with simpleproxy.

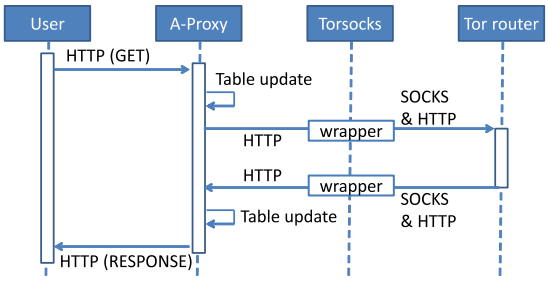


FIGURE 5. The user-proxy-SP sequence with Tor.

The **GM** and **KGC** module are implemented using the TEPLA library [29], with which we implement the Boneh-Franklin IBE scheme and our open-free group signature scheme. The GM module produces a group key and an issuer key, whereas the KGC module produces a public key, and a master secret key. This library supports optimal Ate asymmetric pairings over Barreto-Naehrig (BN) elliptic curves [30] with 254-bit prime order and the corresponding embedded degree is 12. This enables 128-bit security.

C. PERFORMANCE MEASUREMENT

For the purpose of performance measurement, we set up a test system as follows: an Apple MacBookPro (processor: 2.6GHz Intel Core i7, Memory: 16GB, 1600 MHz DDR3, Darwin Kernel Version 13.1.0), and two VMs using VMware Fusion 6.0.2. We assigned the roles of User to the MacOS and the roles of Proxy and SP to each of the VMs. For the Proxy, one of the VMs ran FreeBSD amd64 9.1-RELEASE with one processor and 256MB of memory, and for the SP, the second VM ran CentOS 5.9 x86_64 with one processor and 512MB of memory.

Here, we show that our protocol is feasible by showing the running time of the algorithms and the total running time of one session in the order of msec. First, we show the running time of one session (User→Proxy→SP→Proxy→User) in the following cases: (1) HTTP communication (i.e., without any cryptographic operations), (2) SSL communication, and (3) our protocol in Table 2. To measure the running time of the SSL communication, we use the s_server/s_time command of the OpenSSL library (ver. 1.0.1e) [31]. We use DHE-RSA-AES128-SHA256 cipher suite with a 3,072-bit size public key because this supports 128-bit security similar to ours. Tables show the average time of 1000 times execution.

TABLE 1. Running Time (algorithms).

Algorithm	Entity	Time(msec)	Proxy Module
SendRequest	User	63.90	Simple Proxy
		62.50	Tor
ValidityCheck	SP	87.67	Simple Proxy
		89.40	Tor
SendContent	SP	87.36	Simple Proxy
		85.99	Tor
GetContent	User	52.17	Simple Proxy
		54.23	Tor

TABLE 2. Running time (one session).

Scheme	Cryptographic Operations	Time(msec)	Proxy Module
None	-	2.55	Simple Proxy
	-	8375.48	Tor
SSL	Enc &Auth	14.22	Simple Proxy
		7750.00	Tor
Ours	Enc &Anon. Auth	293.53	Simple Proxy
		9755.53	Tor

First, we compare the running time of each algorithm as follows (see Table 1). Our result proves that each algorithm execution is independent of from the underlying proxy module. Note that the GM.Setup, KGC.Setup, and Join algorithms can be run offline, and that the UserKeyGen algorithm can be run separately against the session. Moreover, we ignore the RelayRequest and RelayContent algorithms because these (run by Proxy) simply relay the communication and are run independently against any cryptographic operations. The dominant factor for the User is the SendRequest algorithm that computes a group signature. Note that this procedure can also be run offline by assuming that the User chooses `TempID` and computes a group signature before starting a session. Then, we can ignore the running time of the SendRequest algorithm.

Next, we show the running time of our protocols as follows. Table 2 demonstrates that the running time of our protocol (Simpleproxy) is approximately 20 times slower than that of SSL communication. This inefficiency is caused by the pairing computation that is not required in usual public key encryption, digital signature, and authentication (these are used in SSL). Nevertheless, it is particularly worth noting that our running time still fits inside the msec order. In addition to this, if the User chooses `TempID` and computes a group signature before starting a session, we can assume that the SendRequest algorithm is run offline. Then, the total running time of one session becomes approximately 200 msec.

Next, we estimate the Tor case. In a Tor network, data are communicated via several Tor routers. That is, different servers are chosen in each communication to hide source IP addresses, and this costs significantly. In reality, the growth of the Tor network was investigated in [32]. Moreover, communication among Tor routers is encrypted (note that this is not an end-to-end encryption, whereas our protocol establishes end-to-end secure channels and therefore,

no Tor routers can reveal content information in our protocol). Because of this, the running time of Tor cases have a wider range at each execution, and from Table and 2, we can interpret that the cost of all cryptographic operations are not dominant when Tor is used as the underlying proxy module.

V. DISCUSSION AND ANALYSIS

This section discusses and analyzes the proposed protocol from the standpoint of compatibility with and deployability over the Internet. It also considers the other approaches to realize secure and anonymous communication.

A. COMPATIBILITY

The proposed protocol can be modified to be compatible with existing HTTP proxies and Tor routers. The current implementation sends an HTTP request using the A-GET method. In this case, only the proxies and Tor routers that understand this extended HTTP request can function as expected, and the other routers cannot manage the request properly.

We intentionally defined the A-GET method instead of using the existing GET method in order to stop communication in case our protocol is not supported by either a Proxy or an SP. If a Proxy or an SP receives a packet with the A-GET method and does not understand the method, it will respond with the status code 400 (Bad Request) and discard the packet, provided that it follows the HTTP protocol. Even if the SP receives the packet, it contains no information that could reveal the identity of the User; thus, anonymity is still maintained.

We could have used the GET method instead of the A-GET method to build anonymous secure communication channels with authentication. In this case, even if the Proxy does not understand our scheme, the packets are relayed to the SP. Thus, the anonymous secure communication channel with authentication is built if the SP understands our protocol, regardless of the Proxy's understanding of our scheme. This could have been alternative approach for designing the protocol, but it completely removes the method for tracking real identity and TempID, i.e., the function that tracks mapping, implemented inside Proxy is disabled.

Moreover, by using the GET method, the User could allow multiple authentication methods with several headers. For instance, the User might employ both an Authentication and A-Authentication header for HTTP basic authentication and our scheme's authentication. In the case where the SP understands our protocol, it runs our scheme's authentication, whereas it runs only the HTTP basic authentication in the case where it does not understand our protocol. In this way, the User can establish a communication channel with or without our scheme depending on SP support of our scheme.

Note that this discussion assumes that the HTTP protocol is implemented properly following RFC 2616 [33], but there is no guarantee that implementations do this properly. Though behavior differs depending on implementation, anonymity is still maintained unless the Proxy intentionally leaks the (IP address, TempID) mapping table to malicious parties.

B. DEPLOYABILITY

In order to use the proposed protocol over the Internet, the User and SP need to manage the proposed protocol. In addition to this, the Proxy needs to be deployed over the Internet. This section discusses the deployability of the Proxy over the Internet.

The Proxy requires several features that are specific to the proposed protocol. Thus we need to implement Proxies over the Internet. The protocol works if we have at least one Proxy over the Internet. This is the same for both Simpleproxy and Tor cases. In the case of Simpleproxy, we need to deploy at least one Simpleproxy over the Internet, so that Users can employ it to run the protocol. In the case of Tor, we need to deploy at least one Simpleproxy that communicates with Torsocks, so that Users can use Tor networks to run the protocol. Thus, the Proxy can be incrementally deployed.

Note that the protocol could have been designed so that no protocol specific features are required for the Proxy, as discussed in Section V-A. In this case, arbitrary HTTP proxies could have been used to run the proposed protocol. Indeed, many HTTP proxies are already available over the Internet, and thus, the protocol can be easily deployed.

C. ALTERNATIVE APPROACH

The proposed protocol realizes secure and anonymous communication but is not the only approach. Another approach is the combined use of our open-free group signature scheme, Tor, and TLS with ephemeral key exchange. It suffices for anonymous communication among parties. Diffie-Hellman (DH) key exchange is also applicable to our model. In this case, a group signature works as a certificate of the public key without revealing the client's identity, gven that the client chooses an ephemeral public key for the DH key exchange, creates a group signature on the key, and sends the signature with the key to the server via proxy entities. This approach is viable pending thorough evaluation and review.

Compared to this approach, our IBE-based approach has an advantage; it incurs smaller costs on the client side in terms of the number of communication sequences. In our protocol, the client computes a group signature on a temporary ID. By contrast, the DH-based protocol requires that the client runs the key exchange protocol in addition to computing a group signature that requires additional interaction and computation. Even if a public key encryption (PKE) scheme is applied, where a client chooses an ephemeral public key and computes a group signature on the key, the client needs to compute the public key from the corresponding secret key. In this case, a secret key needs to be chosen first, following which the corresponding public key is computed (e.g., in the case of ElGamal encryption, a secret key $x \in \mathbb{Z}_p$ is first chosen, and the corresponding public key is then computed, such that $X = g^x$). This requires an additional computation and communication sequence. In case of IBE, no such computation or sequence is required because any value works as the public key.

On the other hand, currently used IBE schemes (e.g., the Boneh-Frankiln scheme that we used) are less efficient in terms of total transaction time, than PKE and DH, even if we take into account the additional cost discussed above. This is because the heavy pairing computation is required to construct an IBE. Indeed, the IBE decryption algorithm requires 50 msec in our implementation whereas the PKE key generation algorithm and its decryption algorithm require only 5 msec in total. Nevertheless, as mentioned in Section IV-C, cryptographic operations are not the dominant factor when Tor is used as the proxy entity. Moreover, the total procedure can be reduced by using IBE as mentioned above. Furthermore, since we have proposed a generic construction of the protocol, any IBE is applicable to it. Thus, when an efficient IBE scheme becomes available in the future, our construction will be rendered more efficient by adopting the scheme.

VI. CONCLUSION

The proposed protocol along with IBE and group signature allow secure anonymous authentication. The difficulty lies in the point where we let encryption and authentication techniques work together without sacrificing anonymity. The proof-of-concept implementation demonstrated the feasibility of the proposed protocol. Based on the implementation, we measured the protocol transaction time and concluded that its performance is within the range of practical acceptance. We also concluded that the protocol is compatible with and deployable over the Internet; although the protocol requires several protocol-specific features, it can draw incremental deployment.

We believe this work can contribute to the management of anonymous communication systems. Assorted anonymous communication systems [3], [34], [35] carry the risk of being used by malicious parties, but they can be sanitized by introducing our protocol and running anonymous user authentication; in this way, illegitimate users cannot use these systems whereas legitimate users can still use them without compromising anonymity. Through this work, we want to facilitate secure, anonymous, and authenticated communication over the Internet.

ACKNOWLEDGMENT

The authors thank Dr. Goichiro Hanaoka and Dr. Miyako Ohkubo for their invaluable comments.

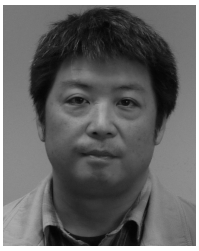
REFERENCES

[1] *Selected Papers in Anonymity*. [Online]. Available: http://freehaven.net/anonbib/date.html, accessed Feb. 20, 2015.
[2] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer-Verlag, 1991, pp. 257–265.
[3] *Tor Project*. [Online]. Available: http://www.torproject.org/, accessed Feb. 20, 2015.
[4] J. Furukawa and H. Imai, "An efficient group signature scheme from bilinear maps," *IEICE Trans.*, vol. E89-A, no. 5, pp. 1328–1338, 2006.
[5] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.
[6] H. Krawczyk, K. G. Paterson, and H. Wee, "On the security of the TLS protocol: A systematic analysis," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer-Verlag, 2013, pp. 429–448.
[7] A. Boldyreva and V. Kumar, "Extended abstract: Provable-security analysis of authenticated encryption in Kerberos," in *Proc. IEEE Symp. Secur. Privacy*, May 2007, pp. 92–100.
[8] J. Wang, G. Wang, and W. Susilo, "Secure single sign-on schemes constructed from nominative signatures," in *Proc. 12th IEEE Int. Conf. TrustCom,* Jul. 2013, pp. 620–627.
[9] A. Sudarsono, T. Nakanishi, Y. Nogami, and N. Funabiki, "Anonymous IEEE802.1X authentication system using group signatures," *J. Inf. Process.*, vol. 18, pp. 63–76, Mar. 2010.
[10] M. Z. Lee, A. M. Dunn, B. Waters, E. Witchel, and J. Katz, "Anon-pass: Practical anonymous subscriptions," in *Proc. IEEE Symp. SP*, May 2013, pp. 319–333.
[11] Y. Gilad and A. Herzberg, "Plug-and-play IP security—Anonymity infrastructure instead of PKI," in *Computer Security—ESORICS*. Berlin, Germany: Springer-Verlag, 2013, pp. 255–272.
[12] J. Sankey and M. Wright, "Dovetail: Stronger anonymity in next-generation internet routing," in *Privacy Enhancing Technologies*. Berlin, Germany: Springer-Verlag, 2014, pp. 283–303.
[13] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, *Internet Key Exchange Protocol Version 2 (IKEv2)*, document Rec. RFC 5996, Sep. 2010.
[14] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1786–1802, Mar. 2011.
[15] K. Emura, A. Kanaoka, S. Ohta, and T. Takahashi, "Building secure and anonymous communication channel: Formal model and its prototype implementation," in *Proc. 29th Annu. ACM Symp. Appl. Comput.*, 2014, pp. 1641–1648.
[16] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Topics in Cryptology—CT-RSA*. Berlin, Germany: Springer-Verlag, 2005, pp. 136–153.
[17] D. Boneh and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," *J. Cryptol.*, vol. 21, no. 2, pp. 149–177, 2008.
[18] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer-Verlag, 1986, pp. 186–194.
[19] B. Libert, T. Peters, and M. Yung, "Group signatures with almost-for-free revocation," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer-Verlag, 2012, pp. 571–589.
[20] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proc. 11th ACM CCS*, 2004, pp. 168–177.
[21] T. Nakanishi and N. Funabiki, "Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer-Verlag, 2005, pp. 533–548.
[22] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki, "Revocable group signature schemes with constant costs for signing and verifying," in *Public Key Cryptography*. Berlin, Germany: Springer-Verlag, 2009, pp. 463–480.
[23] C.-I. Fan, R.-H. Hsu, and M. Manulis, "Group signature with constant revocation costs for signers and verifiers," in *Proc. 10th Int. Conf. CANS*, 2011, pp. 214–233.
[24] N. Attrapadung, K. Emura, G. Hanaoka, and Y. Sakai, "A revocable group signature scheme from identity-based revocation techniques: Achieving constant-size revocation list," in *Proc. 12th Int. Conf. ACNS*, 2014, pp. 419–437.
[25] B. Libert, T. Peters, and M. Yung, "Scalable group signatures with revocation," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer-Verlag, 2012, pp. 609–627.
[26] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.
[27] J. H. Cheon, N. Hopper, Y. Kim, and I. Osipkov, "Provably secure timed-release public key encryption," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 2, 2008, Art. ID 4.
[28] *Simpleproxy: Crocodile Group Software*. [Online]. Available: http://www.crocodile.org/software.html, accessed Feb. 20, 2015.
[29] *TEPLA: University of Tsukuba Elliptic Curve and Pairing Library*. [Online]. Available: http://www.cipher.risk.tsukuba.ac.jp/tepla/index_e.html, accessed Feb. 20, 2015.
[30] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Selected Areas in Cryptography*. Berlin, Germany: Springer-Verlag, 2005, pp. 319–331.

[31] *OpenSSL: Cryptography and SSL/TLS Toolkit*. [Online]. Available: http://www.openssl.org/, accessed Feb. 20, 2015.
[32] M. Edman and P. Syverson, "As-awareness in Tor path selection," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 380–389.
[33] R. Fielding *et al.*, *Hypertext Transfer Protocol–HTTP/1.1*, document Rec. RFC 2616, Jun. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2616.txt
[34] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *Proc. IEEE Symp. SP*, May 2013, pp. 65–79.
[35] H. M. Moghaddam, B. Li, M. Derakhshani, and I. Goldberg, "SkypeMorph: Protocol obfuscation for Tor bridges," in *Proc. ACM CCS*, 2012, pp. 97–108.

KEITA EMURA received the M.E. degree from Kanazawa University, in 2004, and the Ph.D. degree in information science from the Japan Advanced Institute of Science and Technology (JAIST), in 2010. He was with Fujitsu Hokuriku Systems Ltd., from 2004 to 2006. He was with the Center for Highly Dependable Embedded Systems Technology, JAIST, as a Post-Doctoral Researcher, from 2010 to 2012. He has been a Researcher with the National Institute of Information and Communications Technology since 2012, where he has also been a Senior Researcher since 2014. His research interests include public-key cryptography and information security. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), and the International Association for Cryptologic Research. He was a recipient of the SCIS Innovation Paper Award from IEICE in 2012.

AKIRA KANAOKA received the Ph.D. degree in engineering from the University of Tsukuba, in 2004. He was with SECOM Company, Ltd., from 2004 to 2007, and the University of Tsukuba from 2007 to 2013. He is currently an Assistant Professor with Toho University. His research interests include network security and cryptographic application. He is a member of the Institute of Electronics, Information and Communication Engineers, and the Information Processing Society of Japan.



SATOSHI OHTA received the M.S. degree from the Japan Advanced Institute of Science and Technology, in 2002, where he is currently pursuing the Ph.D. degree. He has been a Technical Expert with the National Institute of Information and Communications Technology. His research interests include network protocol and Internet security.

KAZUMASA OMOTE received the Ph.D. degree in information science from the Japan Advanced Institute of Science and Technology (JAIST), in 2002. He was with Fujitsu Laboratories, LTD, from 2002 to 2008, and was involved in research and development of network security. He has been a Research Assistant Professor with JAIST since 2008, where he has also been an Associate Professor since 2011. His research interests include applied cryptography and network security. He is a member of the Institute of Electronics, Information and Communication Engineers, and the Information Processing Society of Japan.

TAKESHI TAKAHASHI (M'05) received the Ph.D. degree in telecommunication from Waseda University, in 2005. He was with the Tampere University of Technology as a Researcher from 2002 to 2004, and Roland Berger Ltd., as a Business Consultant, from 2005 to 2009. Since 2009, he has been with the National Institute of Information and Communications Technology, where he is currently a Senior Researcher. His research interests include Internet security and network protocols. He is a member of the Association for Computing Machinery, and the Institute of Electronics, Information and Communication Engineers. He was a recipient of the Funai Information Technology Incentive Award in 2005, and the ITUAJ's Incentive Award in 2012. He serves as the Co-Chair of IETF MILE Working Group.