IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 27, 2019, accepted May 2, 2019, date of publication May 13, 2019, date of current version June 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916318

ISDSR+: Improving the Security and Availability of Secure Routing Protocol

HIDEHARU KOJIMA, (Member, IEEE), NAOTO YANAI, (Member, IEEE), AND JASON PAUL CRUZ, (Member, IEEE)

Graduate School of Information Science and Technology, Osaka University, Suita 565-0871, Japan

Corresponding author: Naoto Yanai (yanai@ist.osaka-u.ac.jp)

This work was supported in part by the Japan Society for the Promotion of Science KAKENHI under Grant 18K18049 and Grant 18K11262 and in part by the Secom Science and Technology Foundation.

ABSTRACT In ad hoc networks that allow devices to dynamically configure networks via wireless communication, a secure routing protocol is a technology that guarantees the validity of routing with the use of cryptographic authentication. The secure DSR with ID-based sequential aggregate signatures (ISDSR) is a recently introduced secure routing protocol with ''cryptographically compact chain,'' where each device signs both the routing information and signatures generated by the previous device without increasing the size of signatures and the ID information propagated via packets can be utilized as a public key. However, ISDSR requires communication with a centralized key generation center (KGC), and thus a new device may experience difficulties in joining the protocol. Moreover, the implementation results of ISDSR have not been presented. In this work, we present ISDSR+, a new secure routing protocol without a centralized KGC that uses distributed key generation and the conventional features of ISDSR. ISDSR+ relies on a novel signature scheme where any node can receive a secret key as long as the number of available KGCs is more than a certain threshold. In other words, even if several KGCs are unavailable, the remaining available KGCs can still provide secret keys. We furthermore show promising results of ISDSR+ via a prototype implementation on Raspberry Pis. The results show the computational time of 0.1 s for both the secret key generation and the round trip time (RTT) of routing information under reasonable settings. The RTT of ISDSR+ is better than that of a naive secure routing protocol with RSA signatures.

INDEX TERMS Ad hoc networks, distributed key generation, identity-based sequential aggregate signatures, information security, secure routing protocol.

I. INTRODUCTION

A. BACKGROUND

In ad hoc networks, devices such as smartphones and sensors dynamically configure networks in wireless communication without any fixed infrastructure. Therefore, mobile ad hoc networks offer many applications for important and emergency situations, e.g., infrastructure monitoring systems, such as railways [1], landslide detection [2], and underground coals [3], and emergency communication systems during disasters [4]. Furthermore, unmanned aerial vehicles (UAVs) and their flying ad hoc networks (FANETs) have also attracted attention in the recent years [5], [6].

To communicate with a destination device outside its direct transmission area in an ad hoc network, a device exchanges routing information with other devices via a routing protocol, such as AODV [7] and DSR [8], and the packets are forwarded until they reach the destination device. In several protocols [7], [8], a routing information is a concatenation of device identifiers, such as IP address or device name, and a routing protocol requires each device to include its own identifier in the routing information it receives.

Despite its attractive advantages, an ad hoc network typically has weak security because it does not have a fixed infrastructure and it uses dynamic network configuration, i.e., decentralized setting. For example, an adversary can forward packets along another direction that does not lead to the correct destination or loop packets infinitely by manipulating routing information or injecting fake information. These threats are based on an attack against a routing protocol and can often cause serious problems [9]. Such attack can, for

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad S. Khan.

example, prevent the communication of a disaster-stricken user who wants to send an emergency call as soon as possible.

1) SECURE ROUTING PROTOCOLS

Secure routing protocols [10]–[12] prevent the attack described above by guaranteeing the validity of routing information with the use of cryptographic authentication schemes, such as digital signatures. The first secure routing protocol [10] was based on a key management protocol, while subsequent works [11]–[19] adopted digital signatures with public verifiability and non-repudiation. When devices exchange routing information, each device generates a digital signature for the routing information and then sends both the signature and the routing information. Then, another device that receives both the signature and the routing information can check the validity of these information by verifying the signature.

The main difficulty in constructing a secure routing protocol is guaranteeing both the *security and efficiency*, even in large-scale networks. In general, security and efficiency are traded off against each other. The validity of a whole routing information from a source to a destination should be guaranteed, but the introduction of an individual signature per node may decrease performance. Moreover, the security of a routing protocol may not be guaranteed when not all individual signatures from a source to a destination are sent. In several routing protocols [11], [12], digital signatures are exchanged on a part of a whole routing information in networks, but these protocols were later shown to be insecure [20], [21]. However, trivially introducing all individual signatures along with routing information increases packet size in proportion to the number of nodes, and then a packet may be dropped because of its ballooning size. Therefore, the tradeoff between security and efficiency should be solved to design an adequate secure routing protocol.

2) OUR MOTIVATION

The *secure DSR with ID-based sequential aggregate signatures (ISDSR)* [22] has been proposed as a potential solution to the tradeoff problem described above. ISDSR can utilize ID information of each device as a public key, always keep a single short signature independent of the number of devices, and guarantee the whole routing information from any source to a destination. In comparison with other works [12]–[16], [18], [21], [23], ISDSR does not require either the linear size of signatures with respect to the number of hops or the use of public key certificates despite the inclusion of all signatures from a source to a destination. These advantages can be obtained with the use of a state-of-the-art cryptographic scheme named ID-based sequential aggregate signatures [24].

However, ISDSR has two disadvantages. **First, ISDSR requires a centralized key generation center (KGC) to generate a secret key for a requesting node**. Even though ISDSR does not have a fixed infrastructure, a KGC is essentially identical to a fixed infrastructure that a device needs to communicate with to receive a secret key. This kind of communication may become a serious problem particularly during disaster situations, where a device of a user who wants to make an emergency call cannot join a protocol dynamically due to a loss of communication with the KGC. **Second, implementation results of ISDSR with attractive advantages have never been presented**. The effects of decreasing the number of individual signatures and their related information on the performance of the protocol have not been verified. In addition, there are no results that show how the use of state-of-the-art cryptographic schemes improves the performance of the protocol. In our previous work [25], we implemented only the signing algorithm of ISDSR in Java, which greatly increased computational time for signature generation even more than the RSA signatures [26] utilized in the original secure routing protocols [11], [12] with digital signatures. To deploy ISDSR, better results than the use of other existing works should be shown.

Unless these two weaknesses are solved, ISDSR remains incomplete as a secure routing protocol for ad hoc networks.

B. CONTRIBUTIONS

In this work, we present *ISDSR+*, a new secure routing protocol without a centralized KGC, and show its performance via prototype implementation on Raspberry Pis. ISDSR+ is based on a novel signature scheme that allows devices to obtain secret keys as long as the number of available KGCs is more than a certain threshold. The key generation capability of ISDSR+ can be setup in an arbitrary setting, i.e., the number of KGCs and the threshold number for the signature scheme can be chosen arbitrarily. Therefore, ISDSR+ can be considered as a secure routing protocol without any fixed infrastructure. Then, based on implementation results of experiments on Raspberry Pis, we confirm that the computational time of ISDSR+ is faster than that of an RSA-based secure routing protocol, which is a naive-but-fast construction with liner communication costs. Raspberry Pi has been utilized as an actual device for executing ad hoc network protocols, i.e., ISDSR+ and the RSA-based protocol, in our performance evaluation, and thus the results can be used as indicators when an application on an ad hoc network adopts a secure routing protocol utilizing state-of-the-art cryptographic schemes. We have also published our codes in GitHub (https://github.com/naotoyanai/isdsr_plus). The contributions of this work are presented in detail below.

1) ID-BASED SEQUENTIAL AGGREGATE SIGNATURES WITH DISTRIBUTED KEY GENERATION

The first contribution is the removal of a centralized KGC by extending the ID-based sequential aggregate signatures [24] with the introduction of distributed key generation. A device can receive a secret key as long as it can communicate with a number of KGCs more than a threshold designated in advance. In other words, even if several KGCs are unavailable, key generation can still be performed with the remaining accessible KGCs. We call this new scheme *ID-based* *sequential aggregate signatures with distributed key generation (IBSAS-DKG)*. We also show the security analysis of the digital signature algorithm with distributed key generation in a formal proof (See Section IV for details).

2) PROTOTYPE IMPLEMENTATION

The second contribution is the evaluation of the performance of computational time as a round trip time (RTT) by implementing the signature algorithm of ISDSR+ in Java on Raspberry Pis. Our prototype utilizes the JPBC library[1] for the computation of elliptic curves and pairing functions. In comparison with our previous implementation [25], the computational time of the signature algorithm of ISDSR+ is faster by more than 48 times by refactoring with the C language. For our distributed key generation, in the 10-out-of-100 setting, a pair of partial secret key and master public key for a KGC and a secret key for a user can be computed in approximately 330 milliseconds and 100 milliseconds, respectively. Meanwhile, the ISDSR+ has an RTT of 0.1 seconds, which is faster than the RSA-based protocol with RTT of 0.24 seconds. These results show that ISDSR+ can be effectively used in ad hoc networks under reasonable settings (See Section VI for details on our experiments and potential performance).

C. POTENTIAL APPLICATIONS

In addition to the possible application of ISDSR+ in infrastructure monitoring systems [1]–[4] described in Section I-A, it may also be used for securing FANETs. According to the latest survey by Oubbati *et al.* on FANETs [5], FANETs are a special class of mobile ad hoc networks and routing protocols for FANETs are extensions of well-known routing protocols for ad hoc networks, such as the multipath doppler routing (MUDOR) [27] based on DSR and the ad hoc routing protocol for aeronautical MANETs (APRAM) [28] based on AODV. The security of these protocols can be improved by combining them with ISDSR+. Likewise, contents sharing or voice over IP services with UAVs, which are applications of MUDOR and APRAM according to the recent survey by Oubabati *et al.*, are also expected to be potential applications of ISDSR+. The signature verification of ISDSR+ is slightly heavy, and thus we recommend its use in a situation where signatures are verified by only a destination device with powerful computational capability, e.g., a base station for FANETs (See Section VI and Section VII for details on the computational performance and application scenarios, respectively.)

D. PAPER ORGANIZATION

The rest of this paper is organized as follows. Related works are presented in Section II. The model for ISDSR+, research goals, and attack model are presented in Section III. The proposed signature scheme, i.e., IBSAS-DKG, and its deployment to ISDSR+ are discussed in Section IV. Prototype implementation of ISDSR+ is presented in Section V, and then experiments via the implementation are discussed in Section VI. Analysis and evaluation of ISDSR+ are discussed in Section VII. Finally, the conclusion and future direction are presented in Section VIII.

II. RELATED WORKS

In this section, we first describe related works about secure routing protocols. Then, we describe literature on digital signatures, distributed key generation, and security analysis of secure routing protocols.

A. SECURE ROUTING PROTOCOLS

The works by [14], [16], [29] are the closest to our work in terms of the use of signature schemes for multiple signers, where the size of signatures is independent of the number of signers. The motivations of these works are to decrease communication overheads and guarantee the whole routing information. However, they need public key infrastructure while our protocol can achieve infra-less setting. To the best of our knowledge, the first work that presented a secure routing protocol with an attractive cryptographic scheme was the paper in [13]. This protocol and its subsequent work [23] were based on ID-based cryptography [30], which allows any user to utilize any string as a public key. Our protocol contains the advantages of these protocols in addition to the infra-less setting.

Some works presented non-crypto-based routing protocols [31]–[34], but their motivation is the detection of invalid routes and not its prevention. In other words, non-crypto-based routing protocols may still contain risks when routing information is manipulated. Manipulation of routing information, even for a short period, is a serious problem during emergency situations, and thus we focus on secure routing protocols with digital signatures to fully prevent such manipulation.

As further related works to secure routing protocols, Ghosh and Datta [17], [19] proposed addressing schemes with ID-based signatures [35] in a distributed setting, where each node can assign new nodes with IP address dynamically. Although nodes with secret keys in these protocols can dynamically assign with IP addresses, setup of initial nodes is not included. Our proposed protocol can specify the construction of any node with a secret key.

B. DIGITAL SIGNATURES

Among the existing digital signature schemes, schemes where signatures are combined into a single short signature without depending on the number of signers are called *multi-signature schemes* [36]. Multi-signature schemes are classified into two types, i.e., the interactive type [37]–[40], where signers interact with each other to generate a single signature, and the sequential type [24], [41], [42], where signers generate a single signature from a signature-so-far without interaction. The sequential type is more suitable for secure routing protocols, which need signature chains to guarantee the validity of routing information received from nodes.

[1] http://gas.dia.unisa.it/projects/jpbc/

ID-based schemes are more efficient than other schemes, and the sequential type in the ID-based cryptography, i.e., the scheme in [24], is the most suitable scheme for secure routing protocols.

C. DISTRIBUTED KEY GENERATION

Distributed generation of secret keys is suitable for discrete-log-based cryptosystems [43] and there are many such constructions [44]–[46]. The construction of our scheme is close to the threshold signatures proposed by Boldyreva [37].

Related works on distributed key management protocols with ID-based cryptography for ad hot networks [47]–[52] have also been proposed. While ISDSR+ can provide the same capability of distributed key generation as that of the protocols in [47]–[49], the protocols in [50]–[52] can provide key revocation and key update capabilities. Key revocation and key update capabilities can be introduced in ISDSR+ via extensions similar to the protocols in [50]–[52].

D. PROVABLE SECURITY OF SECURE ROUTING PROTOCOLS

Buttyán and Vajda [53] defined a formal security model of ad hoc networks and their routing protocols and analyzed the security of several secure routing protocols [11], [54], [55]. The model by Buttyán and Vajda discussed the security against a single malicious node, and the security was extended by Ács [15] into a model with multiple malicious nodes. Their results show that a secure routing protocol is secure if the digital signature scheme utilized is secure. Kim and Tsudik [14] extended the results of Ács and presented a case utilizing a signature scheme for multiple signers. Our protocol-level security can be guaranteed via the Kim-Tsudik framework [14] although we omit the details due to space limitation.

III. ISDSR+

In this section, we describe a system model of ISDSR+. As described in Section I, ISDSR+ is an extension of the secure DSR with ID-based sequential aggregate signatures (ISDSR). In particular, we introduce distributed key generation in ISDSR+. ISDSR utilizes an ID-based sequential aggregate signature scheme [24], where each user generates a single short signature by taking both messages to be signed and a signature-so-far as input and utilizes any string as its own public key to guarantee routing information sent via packets.

A. PROTOCOL OVERVIEW

ISDSR+ consists of *distributed key generation*, *secure route discovery*, and *secure route maintenance*. Hereinafter, we assume that each node utilizes its own ID information, e.g., IP address or device name, as a public key. In the distributed key generation phase, each node first broadcasts *key request packets (KREQ)* including its own ID information to obtain a secret key. The secret key is computed by multiple KGCs that received the KREQ and then returned to the requesting node by *key reply packets (KREP)*. A node communicates with only a single KGC, but the secret key is generated via interaction between multiple KGCs. In the secure route discovery phase, each node constructs a connection to a destination node by *secure route request packets (SRREQ)* and *secure route reply packets (SRREP)*. In the secure route maintenance phase, the node finds a disconnection to the destination by *secure route error packets (SRERR)*. In both the secure route discovery and the secure route maintenance phases, packets include signatures for routing information between a source and its destination. In ISDSR+, each node that receives these packets from neighbor nodes generates a signature on the received routing information and then adds the generated signature to the packet. We describe each phase in detail below.

1) DISTRIBUTED KEY GENERATION

A node in ISDSR+ needs to obtain a secret key to configure routes to a destination. A node first broadcasts KREQ to KGCs, and then any KGC that receives KREQ responds to the node by returning a global parameter and a master public key for a digital signature scheme. The node communicates with only the first KGC that replies, and this KGC interacts with other KGCs to generate a secret key for the node. Even if several KGCs are unavailable, the remaining KGCs can still generate a key. Finally, the KGC returns a secret key as KREP. From the point of view of a node that broadcasts KREQ, it can obtain a secret key as long as any one of the KGCs is within its direct transmission area.

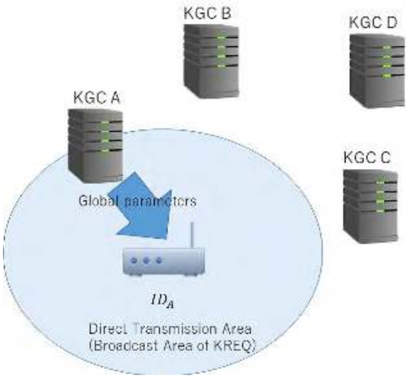

FIGURE 1. Distributed key generation (initial phase)-(a).

Figures 1, 2, and 3 show the procedure for generating a secret key. Figure 1 shows that the node ID_A starts to request key generation to KGCs, and then KGC A responds with the KREQ. Figure 2 shows that KGCs start interacting with one another. Even though KGC D is unavailable, KGC A can generate a secret key with the remaining KGCs, i.e., KGC B and KGC C, and can return the generated secret key as KREP in Figure 3.

2) SECURE ROUTE DISCOVERY

Each node that receives SRREQ adds its own identity in the received routing information and generates a signature for

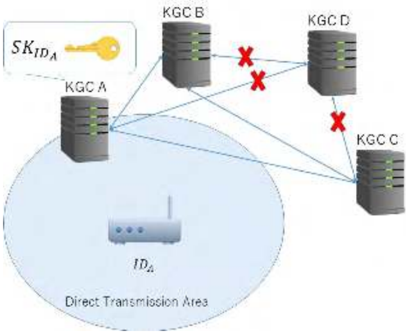

FIGURE 2. Distributed key generation (initial phase)-(b).

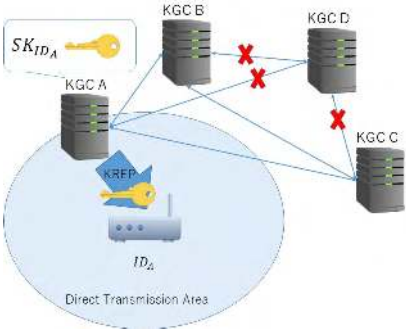

FIGURE 3. Distributed key generation (initial phase)-(c).

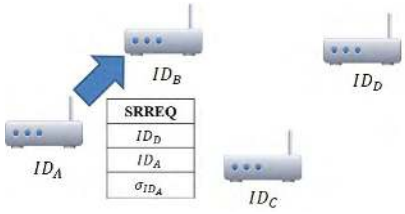

FIGURE 4. Secure route discovery (search phase)-(a).

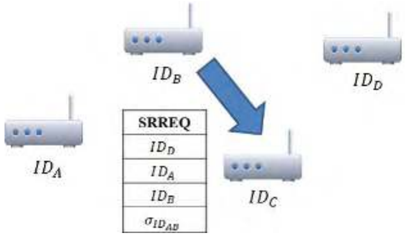

FIGURE 5. Secure route discovery (search phase)-(b).

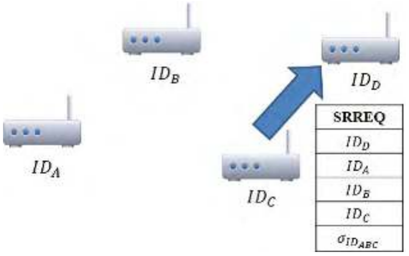

FIGURE 6. Secure route discovery (search phase)-(c).

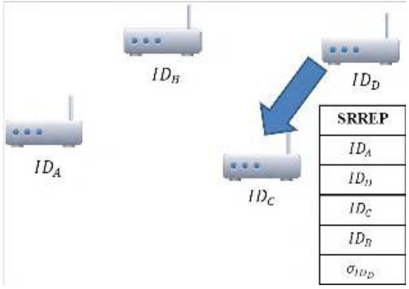

FIGURE 7. Secure route discovery (reply phase)-(a).

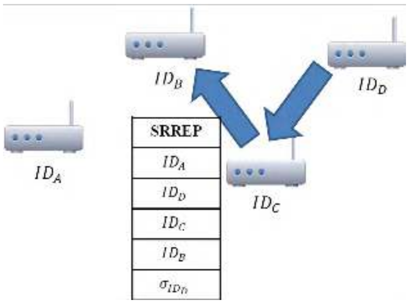

FIGURE 8. Secure route discovery (reply phase)-(b).

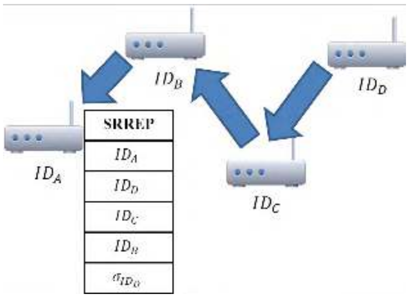

FIGURE 9. Secure route discovery (reply phase)-(c).

a signature algorithm. Then, the node broadcasts SRREQ including the signature. These steps are iterated until a destination node receives packets. The destination node then verifies a signature on SRREQ via a verification algorithm for the underlying signature scheme. If the verification returns True, the destination node generates SRREP by generating a signature on the routing information of SRREQ via the signature algorithm. This SRREP is forwarded to the source node of SRREQ along with the routing information. When the source node receives SRREP, the node verifies the received signature via the verification algorithm. If the verification returns True, the source node registers the given route in its memory.

Figures 4–9 show the procedures for establishing a route. Figures 4–6 show that node ID_A starts to search for a route to the destination node ID_D. In Figure 4, the node ID_A generates a signature σ_{ID_A} via **Signing**, attaches σ_{ID_A} to an SRREQ, and then broadcasts the SRREQ. In Figure 5, the node ID_B receives an SRREQ and generates a signature σ_{ID_B} containing σ_{ID_A}. In Figure 6, the node ID_C does the same process as the node ID_B. After the node ID_D receives a SRREQ from ID_C, the node ID_D verifies the received SRREQ. Figure 7 shows that the node ID_D sends a SRREP to reply to the node ID_A. The signature σ_{ID_D} is generated only by the node ID_D. The node ID_C forwards a received packet, as in Figure 8. In Figure 9, the node ID_A receives a packet SRREP and verifies σ_{ID_D} via **Verification**.

3) SECURE ROUTE MAINTENANCE

When any node (we call such node *ID* for convenience) finds a disconnection to a neighbor on a route registered in its source node, *ID* sends *secure route error packets (SRERR)* including a signature generated via **Signing** to the source. The intermediate nodes between *ID* and the source generate a signature and forward SRERR when they receive the packets. The source node verifies the signature via **Verification** taking the *ID*s as public keys and removes the route if the verification returns True.

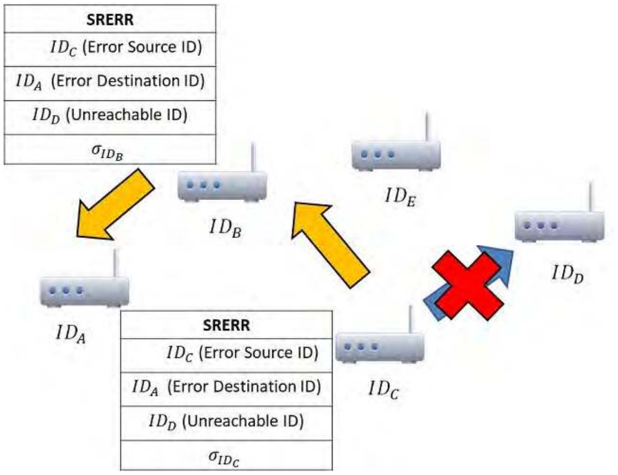

FIGURE 10. Secure route maintenance-(a).

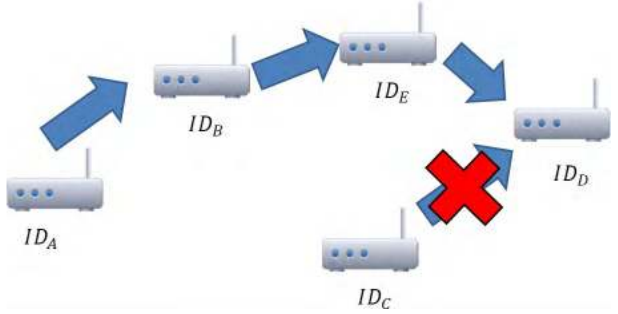

FIGURE 11. Secure route maintenance-(b).

Figures 10 and 11 represent the procedures for secure route maintenance. When the node ID_C detects a disconnection, the node ID_C generates an SREER. The signature σ_{ID_C} is generated by ID_C via **Signing** and ID_B generates a new signature with σ_{ID_C} in Figure 10. After the node ID_A receives an SRERR, it verifies σ_{ID_C} via **Verification** with *ID*s on the route. If the verification passes, ID_A starts to search for other routes to the node ID_D as shown in Figure 11.

B. REQUIREMENTS AND ASSUMPTIONS

We first discuss assumptions used in the proposed model. Networks include several KGCs. For example, in mobile ad hoc networks, base stations can function as KGCs while the nodes of a client cannot become KGCs. We also note that there is no restriction in the ratio of the number of KGCs and the number of nodes of a client in a network. Cryptographic parameters utilized in a digital signature scheme are shared to all KGCs in advance. Each device communicates with other devices wirelessly, and this communication is not encrypted except during an initial phase for generating secret keys.

We now define system requirements for ISDSR+ as follows:

- Unforgeability: An attacker cannot maliciously generate SRREQ, SRREP, and SRERR (We describe the details in the next paragraph).
- Compactness: The size of signatures for SRREQ, SRREP, and SRERR is fixed with respect to the number of hops. Moreover, device-dependent extra information should not be given in the packets.
- Completeness: All signatures (or their related information) have to be given in SRREQ, SRREP, and SRERR to guarantee the validity of routing information for any source and any destination.
- Availability: Any fixed infrastructure should not be used. Moreover, a key generation function should be available even if multiple KGCs become unavailable; unless the number of available KGCs is fewer than a threshold designated in advance.

Unforgeability is the main requirement for secure routing protocols. ISDSR [22] satisfied *Compactness*, *Completeness*, and Unforgeability. In this work, we additionally introduce *Availability* as dynamic configuration of networks for ad hoc networks.

C. ATTACK MODEL

We define an attack model for ISDSR+. In this model, we try to prevent a forgery of routing information with signatures against the malicious forwarding of packets. To do this, we focus on two standpoints, i.e., SRREQ from a source is unforgeable and both SRREP and SRERR to a destination are unforgeable. We assume that a source is always honest since the main motivation of a secure routing protocol is to deliver packets from a source to a destination.

Our target adversary is an *active attacker* [55]. First, we assume the existence of secure channels between honest nodes and KGCs. Precisely, KGCs are fixed before the routing protocol is started, and a secure channel is required only by the closest KGC from a node that requests a secret key. Here, assume that the KGC that a node requests a

Algorithm 1 Setup

```
Ensure: Public Parameter $para$
Generate a pairing parameter $(p, \mathbb{G}, \mathbb{G}_T, e)$.
$P \leftarrow \mathbb{G}$
Choose hash functions $H_1, H_2 : 0, 1^* \rightarrow \mathbb{G}$ and $H_3 : 0, 1^* \rightarrow \mathbb{Z}_p^*$
Set $para = (p, \mathbb{G}, \mathbb{G}_T, e, P, t, \{H_i\}_{i=1,\ldots,3})$
```

secret key from has to be honest.[2] Then, the attacker can eavesdrop, inject, or modify all SRREQ, SRREP, and SRERR and their signatures for any target node adaptively. Moreover, the attacker can setup new nodes in networks, and it can compromise several KGCs. Here, the number of nodes (including KGCs) in a network is not restricted while the number of KGCs compromised by an attacker is assumed to be less than a threshold designated in advance[3]. The attacker then owns all secret keys of the compromised nodes, and it can share the keys among these nodes. The goal of the attacker is to spoof or tamper SRREQ, SRREP, and SRERR. Meanwhile, denial of service attacks and eavesdropping on data sent via the constructed routes are out of the scope of this work because our main target is routing.

Finally, we assume that the attacker is not capable of forging an ID-based sequential aggregate signature scheme with distributed key generation (IBSAS-DKG) described in the next section.

IV. IDENTITY-BASED SEQUENTIAL AGGREGATE SIGNATURES WITH DISTRIBUTED KEY GENERATION

In this section, we propose an ID-based sequential aggregate signature scheme with distributed key generation (IBSAS-DKG) scheme as a new building block of a secure routing protocol. This scheme does not require a centralized KGC but can keep a cryptographically compact chain, as presented in a previous ID-based sequential aggregate signature scheme [24]. We describe only the construction in Algorithms 1–5, and define syntax of IBSAS-DKG and its security in Appendix VIII-A and Appendix VIII-B, respectively. Hereinafter, we assume that each node has a unique index.

A. ALGORITHMS

In this scheme, we utilize bilinear maps and bilinear groups defined as follows. Let $\mathbb{G}$ and $\mathbb{G}_T$ be groups with a common prime order p. A bilinear map $\boldsymbol{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a map such that the following conditions hold:

for all $U, V \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$, $\boldsymbol{e}(aU, bV) = \boldsymbol{e}(U, V)^{ab}$;

for any generator $P \in \mathbb{G}$, $\boldsymbol{e}(P, P) \neq 1_{\mathbb{G}_T}$, $1_{\mathbb{G}_T}$ is an identity element over $\mathbb{G}_T$; and

Algorithm 2 TKeyGen

```
Require: Public Parameter $para$, Threshold $t$, Number $n$ of
    Nodes, Set $\mathcal{D} = [1, n]$ of Nodes, Index $i \in \mathcal{D}$.
Ensure: Master Public Key $mpk$ and Partial Secret Key $x_i$
    for $i$
$(a_{1,i,j}, a_{2,i,j}) \leftarrow \mathbb{Z}_p$ for all $j \in [0, t]$
Set a polynomial function $F_i(x) = \sum_{j=0}^{t} a_{1,i,j}x^j$, where
    $F_i(0) = a_{1,i,0}$
Set a polynomial function $G_i(x) = \sum_{j=0}^{t} a_{2,i,j}x^j$, where
    $G_i(0) = a_{2,i,0}$
Node $i$ sends $F_i(j)$ and $G_i(j)$ to all $j \in \mathcal{D}\backslash\{i\}$
Node $i$ receives $F_i(j)$ and $G_i(j)$ from all $j \in \mathcal{D}\backslash\{i\}$
$a_{1,i} = \sum_{j\in\mathcal{D}} F_j(i)$, $a_{2,i} = \sum_{j\in\mathcal{D}} G_j(j, i)$
Node $i$ sends $(a_{1,i,0}P, a_{2,i,0}P)$ to all $j \in \mathcal{D}\backslash\{i\}$
Node $i$ receives $(a_{1,j,0}P, a_{2,j,0}P)$ from all $j \in \mathcal{D}\backslash\{i\}$
Set $x_i = (a_{1,i}, a_{2,i})$
$(a_1\, P, a_2\, P) = \left(\sum_{j\in\mathcal{D}} a_{1,j,0}P, \sum_{j\in\mathcal{D}} a_{2,j,0}P\right) \in \mathbb{G}^2$
$mpk = (a_1\, P, a_2\, P)$
```

Algorithm 3 UTKeyGen

```
Require: Public Parameter $para$, Master Public Key $mpk$,
    Identity $ID \in \{0, 1\}^*$, Set $\Omega \subseteq \mathcal{D}$ of Nodes such that
    $|\Omega| \geq t$.
Ensure: Secret Key $sk_{ID}$ for $ID$
Send $ID$ to all $j \in \Omega$
For all $j \in \Omega$, $a_{1,j}H_1(ID)$, $a_{2,j}H_2(ID)$
Receive $\left(a_{1,j}H_1(ID), a_{2,j}H_2(ID)\right) \in \mathbb{G}^2$ from all $j \in \Omega$
$a_1H_1(ID) = \left(\sum_{j\in\Omega} \lambda_j a_{1,j}H_1(ID)\right) \in \mathbb{G}$
$a_2H_1(ID) = \left(\sum_{j\in\Omega} \lambda_j a_{2,j}H_2(ID)\right) \in \mathbb{G}$
$sk_{ID} = (a_1H_1(ID), a_2H_1(ID))$
```

Algorithm 4 Signing

```
Require: Public Parameter $para$, Secret Key $sk_{ID_i}$, Message
    $m \in \{0, 1\}^*$, Signature $\sigma$
Ensure: Signature $\sigma'$
Parse $\sigma$ as $(\sigma_1, \sigma_2, \sigma_3)$
If $i = 1$, set $\sigma = (0, 0, 0)$
$(r, x) \leftarrow \mathbb{Z}_p^2$
$\sigma_3' = \sigma_3 + x \cdot P$
$\sigma_2' = \sigma_2 + r \cdot P$
$\sigma_1' \leftarrow \sigma_1 + r\sigma_3 + x\sigma_2' + a_2H_2(ID_i) + H_3(ID_i||m_i)a_1H_1(ID_i)$
```

there is an efficient algorithm that can compute $\boldsymbol{e}(U, V)$ for any $U, V \in \mathbb{G}$.

In this paper, we assume that a discrete logarithm problem (DLP) in $\mathbb{G}$ and $\mathbb{G}_T$ is hard, and say that $\mathbb{G}$ is a bilinear group if all the conditions described above hold. We call the parameter $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ a *pairing parameter*. Algorithm 1 generates a pairing parameter and hash functions $\{H_i\}_{i=1,\ldots,3}$ as global parameters.

Algorithm 2 and Algorithm 3 are the algorithms for distributed key generation. Algorithm 2 is executed to set up

[2] If an attacker can arbitrarily eavesdrop on communication for key generation, the security can be broken easily.

[3] If an attacker can compromise all KGCs without restrictions, then the protocol can be trivially broken.

```
Algorithm 5 Verification
Require: Public Parameter $para$, Master Public Key $mpk$, List $((ID_1, m_1), \ldots, (ID_N, m_N))$ of Identities and Messages, Signature $\sigma$
Ensure: True or False
    Parse $\sigma$ as $(\sigma_1, \sigma_2, \sigma_3)$
    Check if all $ID_1, \ldots, ID_N$ are distinct
    Check $e(\sigma_1, P) = e(\sigma_2, \sigma_3) \cdot e\left(\sum_{i=1}^{N} H_2(ID_i), a_2\ P\right) \cdot e\left(\sum_{i=1}^{N} H_3(ID_i||m_i)H_1(ID_i), a_1\ P\right)$
    return If both checks are true, True
    return If not, False
```

key generation nodes, and Algorithm 3 is executed to create a secret key for each user by the key generation nodes distributively. To instantiate the distributed key generation, we utilize appropriate coefficients by the Lagrange interpolation, i.e., the Lagrange coefficients, and denote the coefficients for some i by $\lambda_j = \prod_{j \in \mathcal{D}} \frac{-i}{j-i}$ for any set $\mathcal{D}$. In Algorithm 2, each key generation node generates polynomial functions (F_i, G_i), where i is an index of the node, and then computes values for other key generation nodes with their indexes, i.e., $F_i(j)$ and $G_i(j)$. By exchanging the values for those computed by the other nodes, each key generation node can obtain a pair of a partial secret key x_i and a common master public key mpk. When a user creates a secret key, it sends its own ID information ID to available key generation nodes, i.e., nodes in a set Ω, and receives partial information $\left(a_{1,j}H_1(ID), a_{2,j}H_2(ID)\right) \in \mathbb{G}^2$ for the secret key for all $j \in \Omega$. If the user can receive partial information more than threshold t, it can obtain the secret key sk_{ID} corresponding to the master public key mpk.

After receiving a secret key sk_{ID}, a user ID can execute Algorithm 4 to generate signatures. In this algorithm, two random numbers (x, r) are generated and then results of their scalar multiplications, i.e., $x \cdot P$ and $r \cdot P$, are added to the components (σ_2, σ_3) of the given signature σ. Then, components related to the secret key and the random numbers are added to the component σ_1. Their resulting signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ can be verified by features of a bilinear map in Algorithm 5.

We show the correctness, i.e., signatures can be verified correctly, in Appendix VIII-C. In addition, the security of the proposed IBSAS-DKG scheme can be proven via formal proofs, which are shown in Appendix VIII-D.

B. DEPLOYMENT OF IBSAS-DKG IN ISDSR+

In this section, we describe how the IBSAS-DKG scheme is deployed in ISDSR+.

First, Algorithm 1 and Algorithm 2 are executed preliminarily, i.e., outside of the routing protocol. When any node needs to obtain a secret key, it sends KREQ to the closest key generation node as a request for the secret key. Then, the key generation node starts Algorithm 3. As long as the key generation node can communicate with other key generation nodes more than threshold t, a secret key can be generated for the requesting node. The resulting secret key is returned to the requesting node as KREP. From the viewpoint of the requesting node, a secret key can always be obtained as long as at least one key generation node is accessible.

Next, SRREQ, SRREP, and SRERR are instantiated by trivially introducing Algorithm 4 in any node except for the use of a timestamp. In particular, when any node tries to discover a route to a destination, it generates a signature using Algorithm 4 with the given signature, where a message to be signed is a concatenation of the ID of a destination node, IDs of intermediate nodes, and a timestamp. The node then forwards the resulting signature instead of the given signature. When a node wants to confirm the validity for any given packet, it can execute Algorithm 5 with the ID list and the signature in that packet.

V. IMPLEMENTATION

In this section, we explain the implementation of three signature algorithms, i.e., Algorithms 1–5, the signature algorithm of ISDSR implemented in our previous paper [25], and the RSA signature algorithm. In our previous paper [25], we implemented the signing algorithm of ISDSR using the JPBC library, in which elliptic curves and pairing functions are written in Java. In this work, we use a PBC library[4] implemented in C language with a Java wrapper provided by JPBC to improve the computational time.

ISDSR+, ISDSR, and RSA-based signature algorithms are implemented in Java as the three applications below. $APP_{ISDSR+-P}$ is implemented with JPBC and uses a Java wrapper API for the PBC native library provided by JPBC. The wrapper plays the role of an interface between JPBC and PBC, and it enables APP$_{ISDSR+-P}$ to execute the computation of elliptic curves and pairing functions with the PBC library. Next, $APP_{ISDSR-J}$ is an implementation of the signature algorithm described in our previous paper [25]. Finally, APP_{RSA} is an implementation for the RSA-based signature algorithm. This application is implemented in Java and its signing and verification functions are executed with the RSA algorithm API of the Java security package.

These three applications have four functions, i.e., signature generation, signature verification, packet management, and sending/receiving packets. The packet management function transforms an instance of a packet class to a byte array to send the packet class data as datagram packets or vice versa. Each application equips an instance of `DatagramSocket` class to send and to receive packets. Figure 12 shows a class diagram of the three applications, where the eight classes are parts of the whole classes of the applications. We explain the `Node` and `AbstractAlgorithm` classes below.

A. CLASSES FOR A NODE

`Node` class is designed for one node that equips a communication socket and a signature algorithm and that has members and methods for sending data, receiving data,

[4] https://crypto.stanford.edu/pbc/

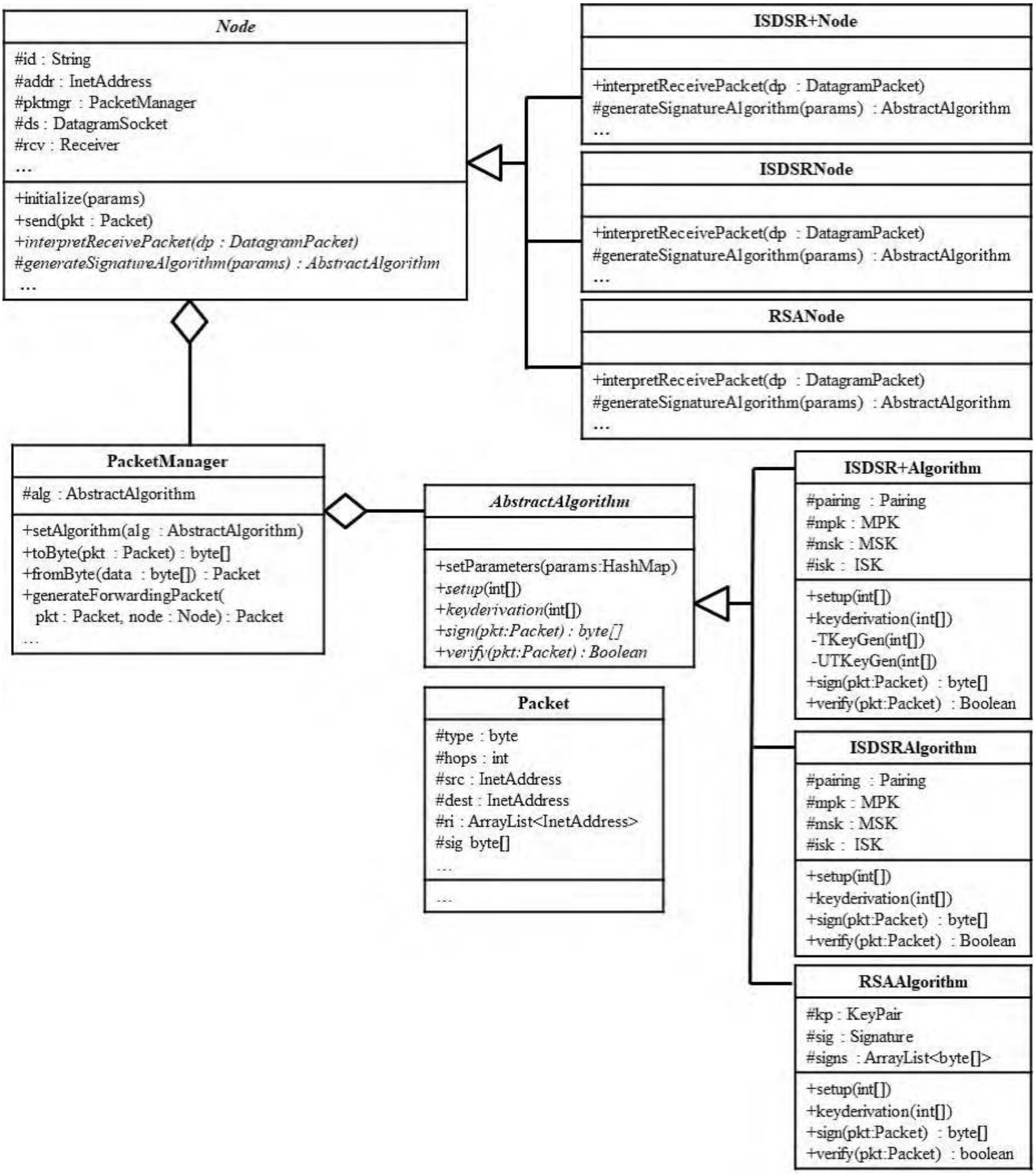
Node
#id : String
#addr : InetAddress
#pktmgr : PacketManager
#ds : DatagramSocket
#rcv : Receiver
...
+initialize(params)
+send(pkt : Packet)
+interpretReceivePacket(dp : DatagramPacket)
#generateSignatureAlgorithm(params) : AbstractAlgorithm
...
ISDSR+Node
+interpretReceivePacket(dp : DatagramPacket)
#generateSignatureAlgorithm(params) : AbstractAlgorithm
...
ISDSRNode
+interpretReceivePacket(dp : DatagramPacket)
#generateSignatureAlgorithm(params) : AbstractAlgorithm
...
RSANode
+interpretReceivePacket(dp : DatagramPacket)
#generateSignatureAlgorithm(params) : AbstractAlgorithm
...
PacketManager
#alg : AbstractAlgorithm
+setAlgorithm(alg : AbstractAlgorithm)
+toByte(pkt : Packet) : byte[]
+fromByte(data : byte[]) : Packet
+generateForwardingPacket(
pkt : Packet, node : Node) : Packet
...
AbstractAlgorithm
+setParameters(params:HashMap)
+setup(int[])
+keyderivation(int[])
+sign(pkt:Packet) : byte[]
+verify(pkt:Packet) : Boolean
Packet
#type : byte
#hops : int
#src : InetAddress
#dest : InetAddress
#ri : ArrayList<InetAddress>
#sig byte[]
...
...
ISDSR+Algorithm
#pairing : Pairing
#mpk : MPK
#msk : MSK
#isk : ISK
+setup(int[])
+keyderivation(int[])
-TKeyGen(int[])
-UTKeyGen(int[])
+sign(pkt:Packet) : byte[]
+verify(pkt:Packet) : Boolean
ISDSRAlgorithm
#pairing : Pairing
#mpk : MPK
#msk : MSK
#isk : ISK
+setup(int[])
+keyderivation(int[])
+sign(pkt:Packet) : byte[]
+verify(pkt:Packet) : Boolean
RSAAlgorithm
#kp : KeyPair
#sig : Signature
#signs : ArrayList<byte[]>
+setup(int[])
+keyderivation(int[])
+sign(pkt:Packet) : byte[]
+verify(pkt:Packet) : boolean

FIGURE 12. A class diagram of the applications.

signing, and verification. A member `rcv`, which is an instance of the `Receiver` class, implements a `Runnable` interface to run as a receiver thread with a member `ds`. An abstract method `interpretReceivedPacket(dp)` is called when `rcv` receives a packet as a `DatagramPacket` from `ds`. The received packet is interpreted through the method `interpretReceivePacket(dp)` implemented in the `ISDSRNode` class and the `RSANode` class. The constructor method calls `generateSignatureAlgorithm (param)` to assign its own signature algorithm to a member `pkitmgr` through a method `PacketManager. setAlgorithm(alg)`. Each method `generate`

SignatureAlgorithm(param) of ISDSR+Node, of ISDSRNode, and of RSANode instantiates an instance of the signature algorithm, such as the ISDSR+Algorithm, the ISDSRAlgorithm, and the RSAAlgorithm, respectively. After each constructor finishes, the member pktmgr has an appropriate instance of the child class of AbstractAlgorithm.

B. CLASSES FOR A SIGNATURE ALGORITHM

The AbstractAlgorithm class is an abstract class for implementing concrete signature algorithm classes, i.e., the ISDSR+Algorithm, the ISDSRAlgorithm, and the RSAAlgorithm, and has four abstract methods, namely, setup(int), keyderivation(int), sign(pkt), and verify(pkt). In the proposed IBSAS-DKG scheme, i.e., the ISDSR+Algorithm, setup(int) corresponds to Algorithm 1, keyderivation(int) corresponds to Algorithms 2 and 3, sign(pkt) corresponds to Algorithm 4, and verify(pkt) corresponds to Algorithm 5. These four methods must be implemented in each concrete class for signature generation and verification using its own signature algorithm.

The ISDSR+Algorithm class applies JPBC version 2.0.0 [56] for computing elliptic curves and pairing functions for signature generation and verification of ISDSR+. JPBC is a port of PBC that performs the mathematical operations underlying pairing-based cryptosystems directly in Java. A constructor method of the ISDSR+Algorithm class instantiates a member pairing to use pairing functions and executes Algorithms 1–3 to generate keys, including a master public key, partial secret keys, and a secret key for each user. The ISDSRAlgorithm class utilizes the JPBC library in which keyderivation(int) is not distributed, i.e., random number generation in a well-known manner. The RSAAlgorithm class uses the RSA algorithm APIs in Java security package, which provides algorithms for signing, verifying, encoding, and decoding. In a constructor method of RSAAlgorithm, members kp and sig are generated from methods in the Java security package.

C. CLASSES FOR A PACKET AND PACKET MANAGEMENT

The Packet class has information to find a route from the source node to its destination node. The PacketManager class has two methods, toByte(pkt) and fromByte(data), to transform a byte array received by a Node.ds to a Packet class instance or vice versa. This class has methods for generating a request packet and a reply packet. When Node class receives a packet, the Node class calls the method pktmgr.generateForwardingPacket to generate a forwarding packet. The method generateForwarding Packet(pkt,node) generates a forwarding packet using a member alg to verify signatures in the pkt or to generate signatures for a new forwarding packet.

VI. EXPERIMENTS

We now evaluate the computational time for distributed key generation and round trip times (hereinafter RTTs) based on our implementation described in the previous section. The execution time of distributed key generation, i.e., KREQ and KREP, is measured on a Raspberry Pi, which is a device that can estimate performances of actual applications by allowing transmissions within real wireless communication, such as Bluetooth and Near-field Communication (NFC). Meanwhile, experiments to measure RTTs between nodes, i.e., SRREQ, SRREP, and SRRER, are conducted to run $APP_{ISDSR+-P}$, $APP_{ISDSR-J}$, and APP_{RSA} on a Raspberry Pi3. The experiments are also conducted in an actual environment including a Raspberry Pi and a desktop PC via LAN connection. In all the experiments described above, we applied "a.properties" in JPBC as parameters to use an elliptic curve and a pairing function in our application program. Here, the bit length of an output group $\mathbb{G}_T$ of a pairing parameter, i.e., a security level, of "a.properties" is 1024 bits. Although such a level is weak in a general usage of a pairing parameter, it is sufficient for secure routing protocols in ad hoc networks. In particular, we can just keep routes secure during a short period when packets are trying to reach their destination.

A. EXPERIMENTAL ENVIRONMENT

We conducted experiments with a Raspberry Pi and a desktop PC described below. The Raspberry Pi is a Raspberry Pi3 model B with Ubuntu 14.4 as OS and Java version of OpenJDK 1.8.1_151. The desktop PC is equipped with Xeon(R) E5-2667 v3 3.20GHz as CPU and 512 GB memory, and it has CentOS 7 as OS and Java version of OpenJDK 1.8.1_151. The desktop PC and the Raspberry Pi connect with a switching hub with a 100Base-T wired LAN.

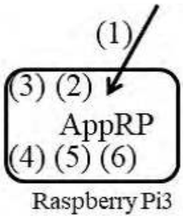

FIGURE 13. A schematic of the procedure for experiments (distributed key generation).

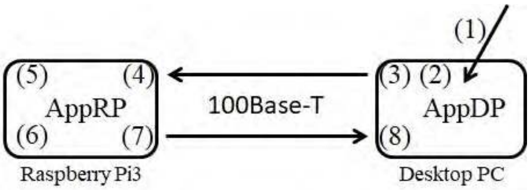

FIGURE 14. A schematic of the procedure for experiments (round trip times).

Figures 13 and 14 show the procedures of an experiment. Hereinafter, we call behaviors of our implemented application program on a desktop PC *AppDP* and those on a Raspberry Pi3 *AppRP*. In Figure 13, the digits (1)–(6) indicate AppRP processes. On the other hand, in Figure 14, the digits (1), (2), (3), and (8) indicate AppDP processes, and the digits (4)–(7) indicate AppRP processes. We describe the behaviors of each experiment below.

First, the behaviors of KREQ and KREP in Figure 13 are described as follows: (1) AppRP is given a value n as the number of KGCs, and then it generates n instances; (2) AppRP generates polynomial functions for each instance; (3) All instances in AppRP distribute values for the polynomial functions with each other; (4) All instances in AppRP generate a partial secret key; (5) Some instances in AppRP generate a master public key; and (6) Some instances in AppRP are given an ID and then generate a secret key for the ID.

The behaviors of SRREQ and SRREP in Figure 14 are described as follows: (1) AppDP is given a value N as the number of nodes; (2) AppDP generates a packet including routing information that contains N; (3) AppDP sends the generated packet; (4) AppRP receives a packet; (5) AppRP verifies signatures in the received packet; (6) After verification passes, AppRP generates its own signatures from the contents of the received packet, and then generates a forwarding packet that includes the signatures and routing information; (7) AppRP sends the generated packet; and (8) AppDP receives a packet and verifies the received packet. The environment described above is common even for SRRER because computational behaviors for each node is almost the same as those of SRREQ and SRREP.

Next, we explain how these experiments are conducted. The experiments in Section VI-B are conducted on a Raspberry Pi without communication between devices even if there are words "send" and "receive" in Algorithm 2 and Algorithm 3. These experiments measure the computational time needed by a real device to generate distributed keys. Consider for example N with value of 50 and t with value of 30. The AppRP creates 50 instances as pseudo nodes. Each pseudo node executes processes from line 1 to line 3 in Algorithm 2 to generate $\mathbf{F}_i(x)$, $\mathbf{G}_i(x)$, $\mathbf{F}_i(0)$, and $\mathbf{G}_i(0)$ as shares. Then, each pseudo node uses shares generated by other instances to execute line 6 in Algorithm 2 to generate $a_{1,i}$ and $a_{2,i}$. Finally, each pseudo node recovers *mpk* by executing line 9 to line 11 in Algorithm 2. These pseudo nodes also execute Algorithm 3 by using $a_{1,j}$ and $a_{2,j}$ from other pseudo nodes.

The experiments in Section VI-C measure the Round Trip Times between the AppDP and the AppRP. For example, when N is 50, the AppDP behaves like the 51st node from a source node because it needs to generate the SRREQ, which includes 50 nodes in its routing information. The AppDP creates 50 instances as pseudo nodes, which are $v_1, v_2, \ldots, v_{50}$, in an application running on the AppDP. First, v_1 generates SRREQ$_1$ by using its own ID and Algorithm 4. Next, v_2 generates SRREQ$_2$ by using its own ID, Algorithm 4, and SRREQ$_1$. The AppDP can get SRREQ$_{50}$ by repeating the processes above until v_{50}. Then, the AppDP can generate SRREQ by using its own ID, Algorithm 4, and SRREQ$_{50}$. Finally, the AppDP can behave like the 51st node from a source node to broadcast the generated SRREQ. When the AppDP gains N with value of 60, it generates SRREQ that contains 61 nodes in its routing information.

B. RESULT OF DISTRIBUTED KEY GENERATION

In this section, we measure computational times for distributed key generation. All results of the experiments are averaged over 10 runs of each application. One experiment represents the case where the AppRP executes both a KREQ and a KREP. Processes that are identical to a KREQ and a KREP are represented by (6) in Figure 13. The other digits represent the case where the AppRP executes threshold key generation of a partial secret key and a master public key, i.e., Algorithm 2.

Figures 15 and 16 show the results of threshold key generation for KGCs and of secret key generation for each user. In particular, Figure 15 show the results of the experiments where the AppRP executed processes (1)–(5) in Figure 13, and Figure 16 show those where the AppDP executed process (6) in Figure 13. The horizontal axis represents threshold values t, and the lines represent different numbers of KGCs n.

The computational time of generating a master key is linear in proportion to t and n as shown in Figure 15, whereas the computational time of generating secret key for a user is quadratic with respect to t as shown in Figure 16. For example, as shown in Figure 15, the computational time of threshold key generation for KGCs can be estimated at $(0.25 * n + 8) * t$ milliseconds in the current environment. On the other hand, the results in Figure 16 are independent of n, and the computational time of secret key generation for a user can be estimated at $(0.09 * t + 1.1) * t$ milliseconds.

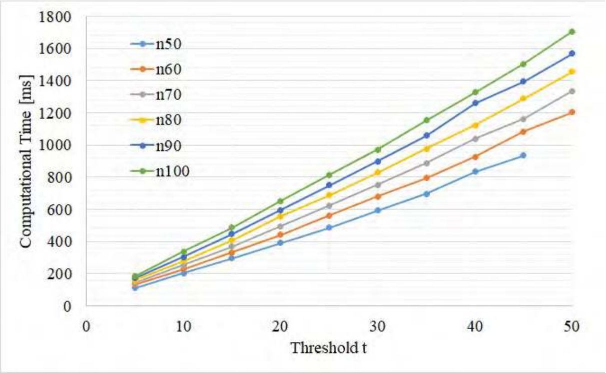

FIGURE 15. **Computational time for distributed key generation: (a) Master key (Algorithm 2).**

Using the estimation described above in a case where $t = 10$ and $n = 100$, the computational time of threshold key generation for KGCs can be estimated to be 330 milliseconds, which is identical to Figure 15. Moreover, the computational time of secret key generation for a user can be estimated to be 19 milliseconds, which is identical to Figure 16.

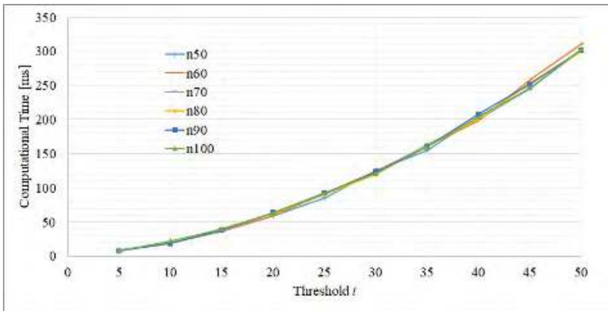

FIGURE 16. Computational time for distributed key generation: (b) User key (Algorithm 3).

C. ROUND TRIP TIME

In this section, we measure round trip times (RTTs) for the route discovery step. All results of the experiments are averaged over 100 runs of each application. One experiment represents the case where the AppRP forwards an SRREQ, which is generated by the received SRREQ without verification. Processes that forward an SRREQ are represented by (4), (6), and (7) in Figure 14. Here, the AppRP does not verify a received packet before the AppRP generates a forwarding SRREQ. The other experiment represents the case where the AppRP replies an SRREP, which is generated by the received SRREQ with verification. Processes that reply the SRREP are represented by (4)–(7) on the AppRP in Figure 14. This experiment indicates that the AppRP is the destination node of the received packet. The AppRP needs to verify the received packet to guarantee routing information in the received packet.

Figures 17–20 show the results when we applied the parameter file "a.properties" to both the APP$_{ISDSR+-P}$ and the APP$_{ISDSR-J}$. Meanwhile, the APP$_{RSA}$ needs the key length as a parameter, and we use 1024 bits, which provides the same security level as that of ISDSR+, for the results in Figures 21 and 22. We also note that the results are also useful for SRRER because its behaviors are almost the same as described above.

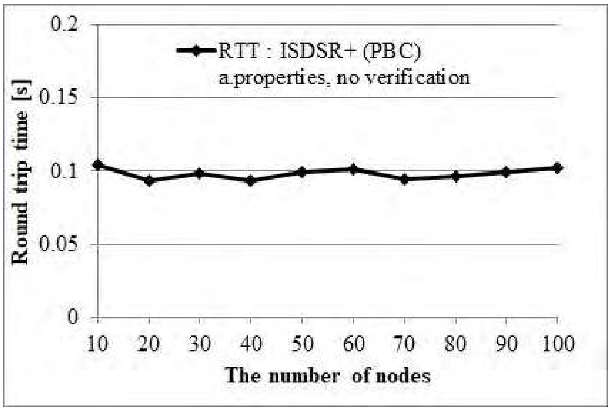

FIGURE 17. RTT of APP$_{ISDSR+-P}$ (PBC) in ISDSR+ against the number of nodes without verification.

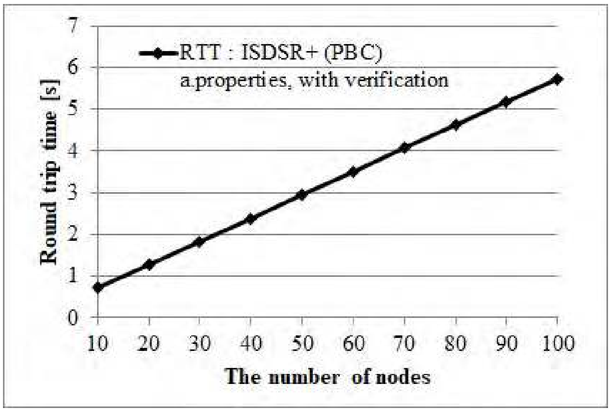

FIGURE 18. RTT of APP$_{ISDSR+-P}$ (PBC) in ISDSR+ against the number of nodes with verification.

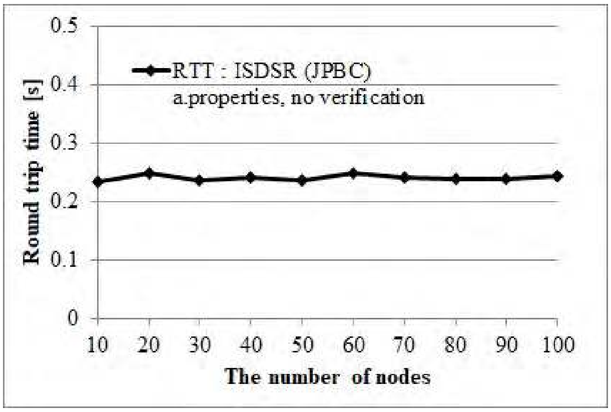

FIGURE 19. RTT of APP$_{ISDSR-J}$ (JPBC) in ISDSR against the number of nodes without verification.

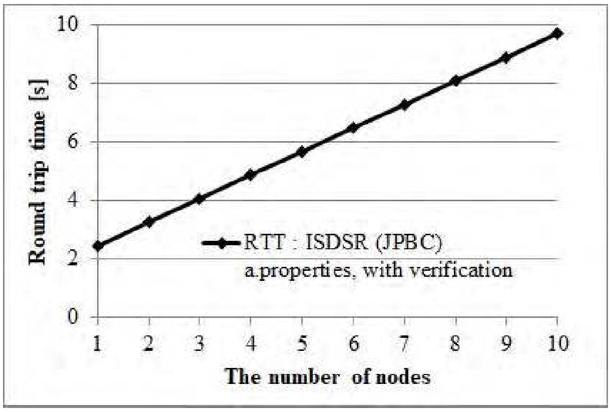

FIGURE 20. RTT of APP$_{ISDSR-J}$ (JPBC) in ISDSR against the number of nodes with verification.

1) CASE: FORWARDING SRREQ

Figures 17, 19, and 21 show results of the experiments where the AppRP executed processes (4), (6), and (7) in Figure 14. The received SRREQ is not verified before generating a forwarded SRREQ. The horizontal axis represents the number of nodes, i.e., the number of nodes in packets sent by the AppDP.

Figure 17 shows the results of APP$_{ISDSR+-P}$, which was computed in the PBC library in C language. Figure 19 shows the results of the APP$_{ISDSR-J}$, which uses JPBC in the generation and verification of signatures.

The RTTs in Figures 17 and 19 are approximately 0.1 seconds and 0.24 seconds, respectively. These results show that the RTTs of ISDSR+ are not dependent on the number of nodes and that the RTTs of the APP$_{ISDSR+-P}$ are faster than those of the APP$_{ISDSR-J}$. Figure 21 shows the results of the APP$_{RSA}$, in which the key length is 1024 bits. Compared with Figure 17, the RTTs in Figure 21 increase when the number of nodes increases. However, the RTTs in Figure 17 are similar at approximately 0.1 seconds even when the number of nodes increases. Based on these results, we can see that ISDSR+ with the PBC library in C language is superior to the RSA-based signature algorithm.

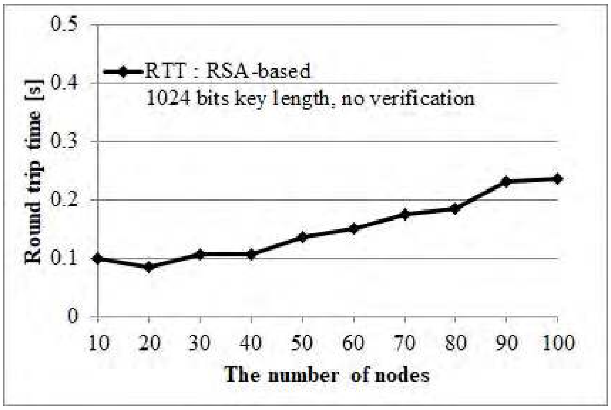

FIGURE 21. RTT of APP$_{RSA}$ against the number of nodes without verification.

2) CASE: REPLYING SRREP

In this case, we assume that the AppRP is the destination node of a received SRREQ. The AppRP has to verify the received SRREQ to guarantee routing information stored in the SRREQ. Then, the AppRP sends the generated SRREP to reply to the source node in the SRREQ. Figures 18, 20, and 22 show the RTTs when the AppRP executes processes (4)–(7) in Figure 14. In Figure 20, we conducted the experiment until 10 nodes because the computational time of APP$_{ISDSR-J}$ is significantly large. Using our proposed implementation, the RTT of APP$_{ISDSR+-P}$ is at least ten times faster than that of APP$_{ISDSR-J}$. In particular, the RTT of APP$_{ISDSR+-P}$ increases by approximately 0.05 seconds with every additional node while that of APP$_{ISDSR-J}$ increases by approximately 0.8 seconds. On the other hand, although The RTT in Figure 22 increases with the number of nodes, the largest RTT is approximately 0.3 seconds.

D. FURTHER POTENTIAL PERFORMANCE

For distributed key generation, a case with a small threshold t and a large number n of whole KGCs is better in a general sense of the availability. The computational time of threshold key generation for KGCs, i.e., Algorithm 2, can be normally separated from configuration of routes from a source to a destination. The computational time of secret key generation for each user, i.e., Algorithm 3, is independent of the number n of KGCs as shown in Section VI-B. We can hence adopt an arbitrary pair of a large n and a small t. For instance, in a case where $t = 5$ and $n = 10000$, a master key for a KGC and a secret key for a user can be computed within approximately 12.4 seconds and 8 milliseconds, respectively. If we use, for example, NFC with its maximum transmission speed of 424 kbit/s or Bluetooth with 24 Mbit/s in the connection between each device and KGCs and 100BASE-T LAN in the connection between the KGCs, transmission of a secret key from the KGCs to a device would take less than 15 milliseconds.

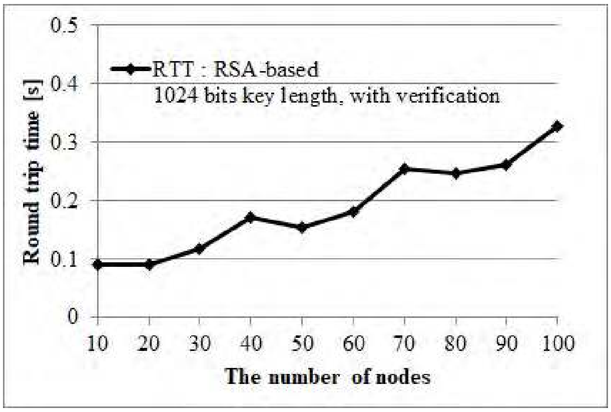

FIGURE 22. RTT of APP$_{RSA}$ against the number of nodes with verification.

The experiments in this paper were conducted on a highly reliable network environment with low latency and low packet drop rate. In this environment, the packets were received completely even when the length of data in a datagram packet was very large. In a practical case where the density of nodes is high, collisions occur easily in the datalink layer. In the datalink layer, the number of frames is large when a packet size is large. A packet sent by the APP$_{RSA}$ needs more frames than that of the APP$_{ISDSR+}$ in the datalink layer because of its packet size. If the frame drop rates of the APP$_{RSA}$ and the APP$_{ISDSR+}$ are the same, then the packet drop rate of the APP$_{ISDSR+}$ is lower than that of the APP$_{RSA}$. Although the APP$_{RSA}$ is superior to the APP$_{ISDSR}$ based on their RTTs, the APP$_{RSA}$ does not work well in unreliable networks because of its large packet size.

We also performed refactoring on our previous implementation to introduce preprocessors for elliptic curve computations in Algorithm 4. Algorithm 4 has six steps, and the steps after step 4 require elliptic curve computations, e.g., $x \cdot P$ and $r \cdot P$. The variable P and a_1 are preprocessed using the API of the JPBC named `jpbc.element.getElementPowPreProcessing()` at the same time when the `ISDSRAlgorithm` class is

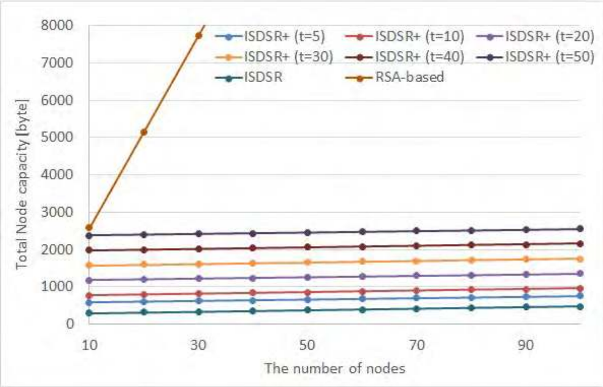

FIGURE 23. Total node capacity for utilizing ISDSR+.

instantiated. The refactoring makes the signing process of ISDSR+ faster.

VII. DISCUSSION

In this section, we briefly discuss how ISDSR+ can achieve requirements described in Section III-B and show potential applications of ISDSR+.

A. ACHIEVEMENT OF REQUIREMENTS

The four requirements described in Section III-B rely on features of IBSAS-DKG. In particular, given its signature compression and the use of IDs for public keys, the compactness and the completeness can be achieved by the guarantee of routing information from any source to any destination. Moreover, by proving the security of IBSAS-DKG and combining it with the Kim-Tsudik framework [14] described in Section II and utilizing a timestamp against replay attacks that reuse valid signatures, the unforgeability of ISDSR+ can be achieved.

Finally, the availability can be achieved by introducing distributed key generation with arbitrary (t, n) setting. Specifically, adopting a large n and a small t enables any device to join the protocol anytime and anywhere. Moreover, any user can receive a secret key by connecting to any KGC within its direct transmission area. We thus conclude that our proposed ISDSR+ can achieve the requirements.

B. NODE CAPACITY AND NUMBER OF KEYS

In this section, we evaluate node capacity for ISDSR+. In particular, we evaluate the size of a memory for each node in terms of the total capacity for utilizing ISDSR+, the capacity for constructing routes in SRREQ and SRREP, the capacity for storing keys, and the capacity during distributed key generation.

The total capacity refers to all the information utilized in ISDSR+, including a secret key, public keys for other nodes to check the validity of packets such as SRREQ, route information constructed by SRREQ and SRREP, and partial information received from each KGC during the distributed key generation, i.e., $a_{1,j}H_1(ID)$ and $a_{2,j}H_2(ID)$.

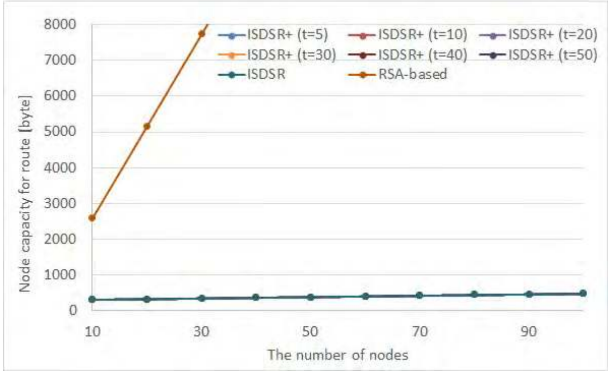

FIGURE 24. Node capacity for constructing routes.

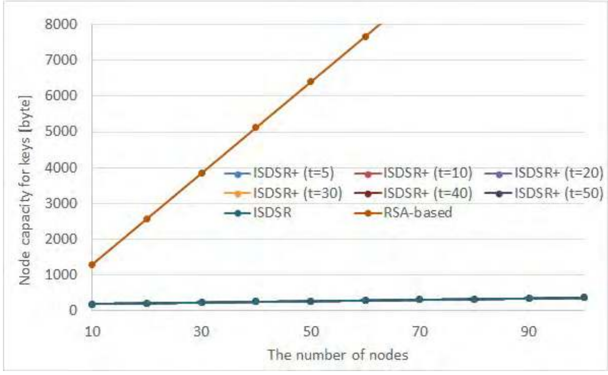

FIGURE 25. Node capacity for storing keys.

The other terms are subsets of total capacity. The capacity for constructing routes in SRREQ and SRREP refers to the information utilized in constructing routes, including a secret key, public keys for other nodes, and route information constructed by SRREQ and SRREP. The capacity for storing keys refers to the size of both a secret key and public keys received from other nodes in SRREQ and SRREP. Finally, the capacity during distributed key generation refers to the total size of the partial information during the distributed key generation, including all the partial information received from each KGC.

Figures 23–26 show the results described above. As can be seen in Figures 24 and 25, the node capacity for ISDSR+ is constant; i.e., the capacity is dependent only on the IDs included in SRREQ and SRREP. As can be seen in Figures 23 and 26, the node capacity for ISDSR+ is linear because the data depend on the number of KGCs in distributed key generation. We note that even if there are 50 KGCs, ISDSR+ is more efficient and effective than the RSA-based protocol. However, ISDSR+ needs a larger memory size than ISDSR because of the distributed key generation.

The discussion on number of keys handles by nodes can be inferred from the discussion on node capacity because keys are necessary for signature verification. ISDSR+ provides only a single master public key with size of 480 bits, consisting of three elements with size of 160 bits each. The key for verifying the signature of a node is identical to the

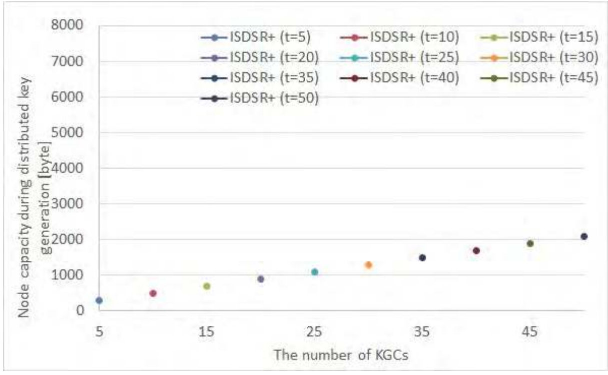

FIGURE 26. Node capacity during distributed key generation.

ID information of the node, eliminating extra information included in the routing protocol. This setting of ISDSR+ is the same as in ISDSR. On the other hand, the RSA-based protocol needs 1048-bit key per node and therefore requires more keys and node capacity than ISDSR+.

C. APPLICATIONS OF ISDSR+

We focus on the RTTs of the applications. By applying the PBC library to the APP$_{ISDSR+-P}$, the RTTs of the APP$_{ISDSR+-P}$ become shorter than those of the APP$_{RSA}$ as long as there are no verification cases. We consider that the signature verification should be executed on a specific destination, e.g., the sink node in sensor networks or a base station in FANETs because they have computational resources that outperform other nodes. For example, the railway monitoring system [1] is a bridge health monitoring system that involves collection of data from wireless sensors installed on a bridge to a remote server for damage identification in the bridge. In this system, a remote server typically has high computational power and can therefore verify packets sent by other sensors.

Another application is content sharing system based on UAVs. As described in Section I-C, according to Oubbati *et al.* [5], several routing protocols [27], [28] for FANETs are based on routing protocols for well-known ad hoc networks and utilized for providing contents to a user. In these applications, UAVs find routes to an access point of a user in conjunction with a request of contents from the user, and then the destination is the access point of the user, who typically owns sufficient computational power.

D. LIMITATIONS

We now describe the limitations of the current results. First, ISDSR+ needs trusted setup for KGCs via the TKeyGen algorithm. There are several theoretical works [57], [58] without any trusted setup for KGCs, and the deployment of such untrusted setup to ISDSR+ remains an open problem.

Another limitation regarding our distributed key generation is that KGCs are fixed before routing protocols. When the number of available KGCs for generation of a secret key of a user, i.e., the use of Algorithm 3, is less than a threshold t, the current ISDSR+ algorithms cannot provide the secret key. These limitations can be solved by introducing evolving secret sharing [59], [60], where a set of parties with secret is not known in advance and could potentially be infinite, in key generation of Algorithm 2. We leave the construction of algorithms via the evolving secret sharing as an open problem.

Another limitation is security against attacks in the physical layer, i.e., jamming attacks. According to the state-of-the-art work in [61], a jamming attack is possible with smartphones and does not require special devices. Several recent works [62]–[64] have proposed data transmission mechanisms that are secure against jamming attacks in the physical layer. The security of ISDSR+ against jamming attacks is not included in our work. Nevertheless, this kind of attack affects distributed key generation and construction of routes, and thus overcoming attacks in the physical aspect is desirable and will be investigated in future works.

VIII. CONCLUSION

In this paper, we proposed a new protocol ISDSR+ by introducing distributed key generation in ISDSR and implementing APP$_{ISDSR+}$ and APP$_{RSA}$ on a Raspberry Pi3. Compared with ISDSR, ISDSR+ does not require a centralized KGC and therefore achieves infra-less setting.

Our main building blocks include a novel signature scheme named IBSAS-DKG, which supports the features of ISDSR+ as described in Section III. We also conducted experiments to measure RTTs. The experiments were classified into without verification and with verification after receiving a packet. In the case of without verification, all results of the APP$_{ISDSR+-P}$ were superior to those of the APP$_{RSA}$.

Finally, as presented in section VII, our experimental environment is more reliable in terms of frame loss rate compared with an environment in practice. Therefore, as future work, we will conduct experiments in an environment that is not highly reliable by using mininet [65] or mininet-wifi [66]. In that case, we will measure the packet delivery ratio between a source node and a destination node. Another future work is the introduction of evolving secret sharing [59], [60] in the distributed key generation of ISDSR+ to overcome the limitations described in Section VII.

APPENDIX
FORMAL DISCUSSION OF IBSAS-DKG

A. SYNTAX

In this section, we define the syntax of an IBSAS-DKG scheme and its correctness.

1) DEFINITIONS OF ALGORITHMS

The algorithms of an IBSAS-DKG scheme are defined as follows. Here, let a message space and an identity space be $\mathcal{M}$ and $\mathcal{ID}$, respectively. We assume that each KGC owns a unique index $i \in \mathbb{N}$, where $\mathbb{N}$ is a set of integers, and a set of KGCs is then defined as a subset of $\mathbb{N}$. We also denote by

$\mathcal{D} \subseteq \mathbb{N}$ any subset of KGCs which own partial secret keys corresponding to a master public key.

Setup: Given a security parameter 1^κ, output a public parameter *para*.

TKeyGen: Given security parameters $(t, n) \in \mathbb{N}^2$, *para*, a set $\mathcal{D} \subseteq \mathbb{N}$ of n KGCs, and an index $i \in \mathcal{D}$, via interaction with all $j \in \mathcal{D} \backslash \{i\}$, output a pair (x_i, mpk) of a partial secret key *msk* as a private output for i and a master public key as a common output for $\mathcal{D}$.

UTKeyGen: Given *para*, *mpk*, *ID* and a set $\Omega \subseteq \mathcal{D}$, output a secret key sk_{ID} corresponding to *ID*.

Sign: Given *para*, a secret key sk_{ID_i}, ID_i, a message $m_i \in \mathcal{M}$ to be signed, a set $L = \{(m_j, ID_j)\}_{j=1}^{i-1}$ of pairs of signed messages and their identities, and an aggregate signature σ, return a new aggregate signature σ' on a new set $L' = L \cup \{(m_i, ID_i)\}$ or $\perp$ to indicate an error.

Verify: Given *para*, *mpk*, a set $L = \{(m_j, ID_j\}_{j=1}^{i}$ of pairs of signed messages and their identities, and an aggregate signature σ, output True or False.

2) CORRECTNESS

The correctness of the IBSAS-DKG scheme described above is defined as follows.

Definition 1 (Correctness): For all para $\leftarrow$ *Setup*(1^κ), *all* $(t, n) \in \mathbb{N}^2$, *all* $ID_i \in \mathcal{ID}$, *all* $m_i \in \mathcal{M}$, *all* $\Omega \subseteq \mathcal{D} \subseteq \mathbb{N}$ *such that* $|\mathcal{D}| = n$ *and* $|\Omega| \geq t$ *hold, all* $L \subseteq \mathcal{M} \times \mathcal{ID}$, *and all* $(x_j, mpk) \leftarrow TKeyGen(t, n, para, \mathcal{D}, j)$ *where* $j \in \mathcal{D}$ *holds, the following condition holds*:

$$True = Verify\left(\begin{array}{c} para, mpk, L', \\ Sign\left(\begin{array}{c} para, \\ UTKeyGen\begin{pmatrix} para, & mpk, \\ ID_i, & \Omega \end{pmatrix}, \\ ID_i, m_i, L, \sigma \end{array}\right) \end{array}\right),$$

where $L' = L \cup \{(m_i, ID_i)\}$. *We say that an IBSAS-DKG scheme is correct if the condition described above holds.*

B. SECURITY DEFINITIONS

In this section, we define security of an IBSAS-DKG scheme. In particular, we define unforgeability of signatures and robustness of distributed key generation as properties of an IBSAS-DKG scheme. The first property is necessary for preventing a signature forgery via received signatures. On the other hand, the second property can prevent an adversary corrupting several KGCs less than a threshold from obtaining the capability of key generation. These two properties are important for the underlying purpose of an IBSAS-DKG scheme, i.e., guaranteeing the validity of routing information and key generation without a fixed infrastructure.

1) UNFORGEABILITY

We define the unforgeability of an IBSAS-DKG scheme via the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Namely, an advantage of $\mathcal{A}$ can be obtained with a probability that $\mathcal{C}$ outputs *accept* in the game. Hereinafter, we denote by $x^{(i)}$ a value of the i-th query for all x.

Initial Phase: The challenger $\mathcal{C}$ generates a public parameter *para* $\leftarrow$ *Setup*(1^κ), and chooses integers $(t, n) \in \mathbb{N}^2$ and a set $\mathcal{D} \subseteq \mathbb{N}^n$ of n KGCs. Next, $\mathcal{C}$ generates $(x_i, mpk) \leftarrow TKeyGen(t, n, para, \mathcal{D}, i)$ for all $i \in \mathcal{D}$. $\mathcal{C}$ then runs $\mathcal{A}$ with $(t, n, para, mpk, \mathcal{D})$ as input.

Corrupt Query: $\mathcal{A}$ sends any index $i^{(h)} \in \mathcal{D}$ to $\mathcal{C}$, and $\mathcal{C}$ returns a partial secret key $x_{i^{(h)}}$ for the given index $i^{(h)}$.

KeyDer Query: $\mathcal{A}$ sends any string $\left(ID^{(h)}, \Omega \subseteq \mathcal{D}\right)$ to $\mathcal{C}$, and $\mathcal{C}$ returns a secret key $sk_{ID^{(h)}}$ for $ID^{(h)}$.

Sign Query: $\mathcal{A}$ generates a signing query $(para, mpk, m^{(h)}, ID^{(h)}, L, \sigma)$. Given the query, $\mathcal{C}$ returns a signature σ.

Output After q_c iterations of the **Corrupt Query**, q_k iterations of the **KeyDer Query**, and q_s iterations of the **Sign Query**, $\mathcal{A}$ outputs a forgery (L^*, σ^*), where $L = \{(ID_i^*, m_i^*)\}_{i=1}^{N}$ where $N \in \mathbb{N}$ and the following conditions hold: the **Verify** algorithm outputs True; $q_c < t$ holds; there is exactly one $ID_{i^*}^*$ such that $ID_{i^*}^* \notin \{ID_i^{(h)}\}_{h=1}^{q_k}$ holds for the **KeyDer Query**; for the $ID_{i^*}^*$, a tuple of $(m_{i^*}^*, ID_{i^*}^*, L_{i^*-1}^* = \{(ID_j^*, m_j^*)\}_{j=1}^{i^*-1})$ has never been queried to the **Sign Query**; and all $ID^* \in L^*$ are distinct. If all the conditions hold, then $\mathcal{C}$ outputs *accept*. Otherwise, $\mathcal{C}$ outputs *reject*.

Definition 2: We say that an IBSAS-DKG scheme is $(t, n, q_c, q_k, q_s, q_h, N, \epsilon)$*-unforgeable if there is no probabilistic polynomial-time adversary* $\mathcal{A}$ *who forges with* $(t, n, q_c, q_k, q_s, q_h, N, \epsilon)$. *Here, we say that* $\mathcal{A}$ *forges the scheme with* $(t, n, q_c, q_k, q_s, q_h, N, \epsilon)$ *if a challenger* $\mathcal{C}$ *outputs accept, in the security game described above, with a probability greater than* ϵ *on a threshold t and a number n of KGCs. Here,* $\mathcal{A}$ *can generate at most* q_c *corruption queries, at most* q_k *key derivation queries, at most* q_s *signing queries, and at most* q_h *random oracle queries, and N is the number of signers in the* $\mathcal{A}$*'s output and queries.*

2) ROBUSTNESS

We define the robustness via the following simulation between a real scheme and an ideal scheme for an adversary $\mathcal{A}$. Specifically, a challenger $\mathcal{C}$ executes either the algorithms defined in Section A or algorithms without threshold computations. Then, the robustness is defined as follows:

Definition 3: We say that an IBSAS-DKG scheme is robust if for all mpk there are ***KeyGen*** *and* ***UKeyGen*** *whose distributions are identical for any probabilistic polynomial-time algorithm* $\mathcal{A}$ *to* ***TKeyGen*** *and* ***UTKeyGen****, respectively. Here,* $\mathcal{A}$ *can corrupt up to* $t - 1$ *signers in order to obtain* $t - 1$ *partial secret keys.*

C. CORRECTNESS OF THE SCHEME

In this section, we briefly show that our scheme in Section IV is correct in the meaning of A2. In particular, we first check if a secret key generated by the **UTKeyGen** algorithm is identical to a part of the **Verify** algorithm, and then show that the **Verify** algorithm outputs True via the resultant secret key.

Theorem 1: The proposed scheme is correct.

Proof: First, for any ID and its related secret key, the following equations hold via the Lagrange interpolation:

$$\begin{aligned} e\left(\sum_{j\in\Omega}(\lambda_j\alpha_{1,j}H_1(ID)),P\right) &= e\,(a_1\ H_1(ID)),P) \\ &= e\,(H_1(ID)),a_1\ P)\,, \\ e\left(\sum_{j\in\Omega}(\lambda_j\alpha_{2,j}H_2(ID)),P\right) &= e\,(a_2\ H_2(ID)),P) \\ &= e\,(H_2(ID)),a_2\ P)\,. \end{aligned}$$

For the **Verify** algorithm, the verification equation can be written as follows: for any i, $n(\geq i)\in\mathbb{Z}^2$:

$$\begin{aligned} e(\sigma_1,P) &\overset{?}{=} e\left(\begin{matrix} rxP+\sum_{i=1}^N a_2H_2(ID_i) \\ +\sum_{i=1}^N H_3(ID_i||m_i)a_1H_1(ID_i)\end{matrix},P\right) \\ &= e(rxP,P)\cdot e\left(\sum_{i=1}^N a_2H_2(ID_i),P\right) \\ &\quad\times e\left(\sum_{i=1}^N H_3(ID_i||m_i)a_1H_1(ID_i),P\right) \\ &= e\,(xP,rP\ )\cdot e\left(\sum_{i=1}^N H_2(ID_i),a_2\ P\right) \\ &\quad\times e\left(\sum_{i=1}^N H_3(ID_i||m)H_1(ID_i),a_1\ P\right). \end{aligned}$$

The computation step described above is identical to the verification equation on the **Verify** algorithm. The algorithm then returns True. The proposed scheme is thus correct because the two conditions described in Section A2. □

D. SECURITY ANALYSIS OF IBSAS-DKG

We analyze the security of the proposed scheme by formal proofs. We first prove that the scheme is unforgeable under the security assumptions named ID-based sequential aggregate signature computational Diffie-Hellman (IBSAS-CDH) assumption [24] defined below. Next, We prove that the scheme is robust in the sense of the definitions in Section B.

Definition 4 ((q,ϵ)-IBSAS-CDH Assumption in $\mathbb{G}$): We define an IBSAS-CDH problem with a security parameter 1^k as follows: for a pairing parameter $(p,\mathbb{G},\mathbb{G}_T,\boldsymbol{e})$ and a given tuple (P,a_1P,a_2P,b_1P,b_2P) with uniformly random $(a_1,a_2,b_1,\ b_2)\leftarrow\mathbb{Z}_p^4$ as input compute $(rxP{+}a_1\ b_1P{+}ma_2\ b_2P,\ \ rP,xP)$ for uniformly random $(r,x)\leftarrow\mathbb{Z}_p$ under accessing to an oracle $\mathcal{O}^{IBSAS-CDH}_{P,a_1P,a_2P,b_1P,b_2P}$ that takes $m\in\mathbb{Z}_p$ as input and returns $(rxP{+}a_1\ b_1P{+}ma_2\ b_2P,rP\ \ ,xP)$ for randomly generated numbers $(r,x)\leftarrow\mathbb{Z}_p^2$, where an element m involved in each query must be different from the element m involved in the final output. We say that a (q,ϵ)-IBSAS-CDH assumption in $\mathbb{G}$ holds if there is no probabilistic polynomial-time algorithm that can solve the IBSAS-CDH problem with a probability greater than ϵ. Here, the algorithm can generate at most q queries to the oracle.

Theorem 2: Suppose that H_i for $i\in[1,3]$ is modeled as a random oracle. The proposed scheme is $(t,n,q_c,q_j,q_k,q_s,q_r,q_{h_1},q_{h_2},q_{h_3},N,\epsilon)$-unforgeable under the (q,ϵ')-IBSAS-CDH assumption in $\mathbb{G}$, where $q\leq q_s$,

$$\epsilon'\geq\epsilon\left(1-\frac{q_c+1}{n}\right)\frac{1}{e(N(q_s+1)+q_k+1)},$$

e is the base of the natural logarithm.

Proof: Given $(p,\mathbb{G},\mathbb{G}_T,\boldsymbol{e},P,a_1P,a_2P,b_1P,b_2P)$ and access to an oracle $\mathcal{O}^{IBSAS-CDH}_{P,a_1P,a_2P,b_1P,b_2P}$, $\mathcal{B}$ sets $(p,\mathbb{G},\mathbb{G}_T,\boldsymbol{e},P)$ as *para* and $(a_1\ P,a_2P)$ as *mpk*. Next, $\mathcal{B}$ sets $\mathcal{ID}[-,-,-]$, $\mathcal{D}[-,-,-,-,-]$, $H_1[-,-,-]$, $H_2[-,-,-]$ and $H_3[-,-]$ as an $\mathcal{ID}$-list, $\mathcal{D}$-list, a H_1-list, a H_2-list, and a H_3-list, respectively. Then, $\mathcal{B}$ guesses some index $i^*\in[1,n]$. For $i\in[1,n]\backslash\{i^*\}$, $\mathcal{B}$ chooses polynomial functions $F_i(x)$ and $G_i(x)$ with the degree t, and computes $a_{1,i}$, $a_{2,i}$, $a_{1,i,0}P$ and $a_{2,i,0}P$. $\mathcal{B}$ then computes $a_{1,i^*,0}P=a_1\ P-\sum_{i\in[1,n]\backslash\{i^*\}}a_{1,i,0}$ and $a_{2,i^*,0}P=a_2\ P-\sum_{i\in[1,n]\backslash\{i^*\}}a_{2,i,0}$ as a partial secret key x_i. Next, $\mathcal{B}$ registers $(ID_i,F_i(x),G_i(x),a_{1,i},a_{2,i})$ in the $\mathcal{D}$-list for $i\in[1,n]\backslash\{i^*\}$, and registers $(ID_{i^*},-,-,-,-)$ in the $\mathcal{D}$-list for i^*. $\mathcal{B}$ sets $\mathcal{D}=[1,n]$, and runs $\mathcal{A}$ with $(p,\mathbb{G},\mathbb{G}_T,\boldsymbol{e},P,a_1P,a_2P,b_1P,b_2P,\mathcal{D})$ in the following manner. Here, $\mathcal{B}$ utilize a random coin $B\in\{0,1\}$ with a probability ε to set 1, and we finally determine ε to complete the proof:

H_1 **Query** $(ID_i^{(h)})$**:** This oracle simulation is associated with the H_2 **Query**. If $(ID_i^{(h)})$ has been registered in the H_2 list, $\mathcal{B}$ retrieves B_i from the list. Otherwise, $\mathcal{B}$ sets $B_i=1$ with the probability ε or $B_i=0$ with the probability $1-\varepsilon$. Next, $\mathcal{B}$ generates $\alpha_i\leftarrow\mathbb{Z}_p^*$, and then sets $H_1(ID^{(h)})=\alpha_iP$ for $B_i=1$ or $H_1(ID^{(h)})=\alpha_iP+b_1\ P$ for $B_i=0$. $\mathcal{B}$ registers $(ID_i^{(h)},\alpha_i,B_i)$ in the $\mathcal{H}_1$-list, and returns the $H_1(ID^{(h)})$.

H_2 **Query** $(ID_i^{(h)})$**:** This oracle simulation is associated with the H_1 **Query**. If $(ID_i^{(h)})$ has been registered in the H_1 list, $\mathcal{B}$ retrieves B_i from the list. Otherwise, $\mathcal{B}$ sets $B_i=1$ with the probability ε or $B_i=0$ with the probability $1-\varepsilon$. Next, $\mathcal{B}$ generates $\beta_i\leftarrow\mathbb{Z}_p^*$, and then sets $H_2(ID^{(h)})=\beta_iP$ for $B_i=1$ or $H_2(ID^{(h)})=\beta_iP+b_2\ P$ for $B_i=0$. $\mathcal{B}$ registers $(ID_i^{(h)},\beta_i,B_i)$ in the $\mathcal{H}_2$-list, and returns the $H_2(ID^{(h)})$.

H_3 **Query** $(ID_i^{(h)}\parallel m_i^{(h)})$**:** $\mathcal{B}$ generates $\delta_i\leftarrow\mathbb{Z}_p^*$ and sets $H_3(ID_i^{(h)}\parallel m_i^{(h)})=\delta_i$. $\mathcal{B}$ then registers $(ID_i^{(h)}\parallel m_i^{(h)},\delta_i)$ in the $\mathcal{H}_3$-list, and returns the $H_3(ID_i^{(h)}\parallel m_i^{(h)})$.

Corrupt Query $(i^{(h)})$**:** $\mathcal{B}$ first checks if $i^{(h)}=i^*$. If so, $\mathcal{B}$ aborts the process. Otherwise, $\mathcal{B}$ retrieves a partial secret key $x_i=(a_{1,i},a_{2,i})$ from the $\mathcal{D}$-list, and returns x_i.

KeyDer Query $\left(ID^{(h)},\Omega\subseteq\mathcal{D}\right)$**:** $\mathcal{B}$ checks if $ID^{(h)}$ has been registered in the H_1-list and the H_2-list. If not, $\mathcal{B}$ executes H_1 **Query** and H_2 **Query**. Next, $\mathcal{B}$ checks if $B_i=0$ on the lists for $ID^{(h)}$. If so, $\mathcal{B}$ aborts the process. If not, i.e., $B_1=1$, $\mathcal{B}$ computes $sk_{ID^{(h)}}=(\alpha_ia_1P,\beta_ia_2\ P)$. This $sk_{ID^{(h)}}$ can be written as $a_1H_1(ID^{(h)})=\alpha_ia_1P$ and $a_2H_2(ID^{(h)})=\beta_ia_2\ P$ from the H_1-list and the H_2-list, respectively. Finally, $\mathcal{B}$ registers $\left(ID^{(h)},a_1H_1(ID^{(h)}),a_2H_2(ID^{(h)})\right)$ in the $\mathcal{ID}$-list, and returns a secret key $sk_{ID^{(h)}}$.

Sign Query $(para, mpk, m_i^{(h)}, ID_i^{(h)}, L, \sigma)$**:** $\mathcal{B}$ checks if $B_i = 1$ for $ID_i^{(h)}$. If so, $\mathcal{B}$ executes **KeyDer Query** and then generates a signature in the same manner as that of the original *Sign* algorithm. This distribution is exactly identical to that of the original algorithm except for the use of random oracles because $\mathcal{B}$ can know a secret key.

Otherwise, i.e., $B_i = 1$ for $ID_i^{(h)}$, $\mathcal{B}$ checks if L contains ID_j for $j \in [1, i-1]$ such that $B_j = 0$ holds on the H_1-list and the H_2-list. $\mathcal{B}$ aborts the process if so. Otherwise, $\mathcal{B}$ discards the given σ, and retrieves δ_i for $ID_i^{(h)} \parallel m_i^{(h)}$ from the H_3-list. $\mathcal{B}$ then receives $(rxP + a_1\ b_1P + \delta_i a_2\ b_2P,\ rP\ ,\ xP)$ by accessing to the oracle $\mathcal{O}^{IBSAS-CDH}_{P,a_1P,a_2P,b_1P,b_2P}$ with δ_i as input. $\mathcal{B}$ computes $rxP + a_1\ b_1P + \delta_i a_2\ b_2P + \sum_{j=1}^{i} \left(\alpha_j a_1 P + \delta_j \beta_j a_2\ P\right)$ as σ_1 by retrieving α_j and β_j for all $j \in [1, i]$ from the H_1-list and the H_2-list. $\mathcal{B}$ also sets $\sigma_2 = rP$ and $\sigma_3 = xP$. This σ_1 can be written as $rxP + \sum_{j=1}^{i} \left(a_1 H_1(ID_j) + a_2\ H_3(ID_j \parallel m_j) H_2(ID_j)\right)$ from simulation of each list. The signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is accepted on the verification algorithm, and thus its distribution in the simulation described above is indistinguishable for $\mathcal{A}$. $\mathcal{B}$ returns a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$.

Output: After the simulation described above, $\mathcal{A}$ outputs a forgery (L^*, σ^*) where there exists exactly a single $ID_{j^*}^*$, whose secret key has never been queried to **KeyDer Query**, from the definition of the forgery. $\mathcal{B}$ checks if, for $ID_{j^*}^*$, $B_{j^*} = 0$ holds on the H_1-list and the H_2-list and, for other $ID_j^* \in L^*$, $B_{j^*} = 1$ holds. If the statement is false, $\mathcal{B}$ aborts the process. Otherwise, the signature can be written as follows because the verification holds:

$$\sigma_1 = rxP + a_1\ b_1P + \delta_{j^*} a_2\ b_2P + \sum_{i=1}^{n} (a_1\alpha_i P + \delta_i a_2 \beta_i P)\ .$$

$\mathcal{B}$ can hence extract a solution to the problem by following computation:

$$\begin{aligned}\sigma' &= \sigma_1 - \sum_{i=1}^{n} (\alpha_1\ a_1 P + \delta_i \beta_i a_2 P)\\ &= rxP + a_1\ b_1\ P + \delta_{j^*} a_2\ b_2\ P,\end{aligned}$$

where δ_{j^*} is a value which has never been queried to the oracle. The tuple of $(\sigma', \sigma_2, \sigma_3)$ is a solution to the IBSAS-CDH problem.

To complete the proof, we analyze a success probability ϵ' of $\mathcal{B}$. In addition to a success probability ϵ of $\mathcal{A}$, there are five cases where $\mathcal{B}$ aborts the process; the first case $abort_C$ is in the **Corrupt Query** where a partial secret key for i^* has been queried; the second case $abort_K$ is in the **KeyDer Query** where a secret key with $B_i = 0$ in the H_1-list or the H_2-list has been queried; the third case $abort_S$ is in the **Sign Query** where there are multiple identities with $B_i = 0$ in the H_1-list or the H_2-list for any i; and, the firth case $abort_{out}$ is in the **Output** where the statement about B_{j^*} is false. The success probability can be then estimated as follows:

$$\begin{aligned}\epsilon' &= \epsilon \cdot \Pr[\bigwedge_{j=1}^{q_c} \neg abort_C] \cdot \Pr[\bigwedge_{j=1}^{q_k} \neg abort_K]\\ &\quad \times \Pr[\neg abort_S] \cdot \Pr[\neg abort_{out}]\\ &\geq \epsilon \cdot \left(\frac{n-1}{n} \cdot \frac{(n-1)-1}{n-1} \cdots \frac{(n-q_c)-1}{(n-q_c)}\right) \cdot \left(\varepsilon^{q_k}\right)\\ &\quad \times \left(\varepsilon^{Nq_s}\right) \cdot \left(\varepsilon^{(N-1)}(1-\varepsilon)\right)\\ &\geq \epsilon \cdot \left(\frac{(n-q_c)-1}{n}\right) \cdot \varepsilon^{q_k+Nq_s+N-1}(1-\varepsilon)\\ &\geq \epsilon \cdot \left(1 - \frac{q_c+1}{n}\right) \cdot \varepsilon^{q_k+N(q_s+1)}(1-\varepsilon).\end{aligned}$$

The variable ε is finally determined in order to optimize the probability described above. Here, let $f(\varepsilon)$ be a function $\varepsilon^z(1-\varepsilon)$ where $z = q_k + N(q_s+1)$. Then, $f(\varepsilon)$ is maximized at $\varepsilon_{opt} := \frac{z}{z+1}$ according to the derived function. That is, the following inequation can be obtained for the function $f(\varepsilon)$.

$$\begin{aligned}f(\varepsilon_{opt}) &= \left(\frac{z}{z+1}\right)^a \left(1 - \frac{z}{z+1}\right)\\ &= \left(1 + \frac{1}{z}\right)^{-z} \left(\frac{1}{z+1}\right) \geq e^{-1} \left(\frac{1}{z+1}\right),\end{aligned}$$

where e is the base of the natural logarithm. The success probability ϵ' is then bounded as follows:

$$\epsilon' \geq \epsilon \cdot \left(1 - \frac{q_c+1}{n}\right) \cdot \left(\frac{1}{e(q_k + N(q_s+1)+1)}\right).$$

The probability is polynomially bounded.

□

Theorem 3: The proposed scheme is robust.

Proof: In this proof, we show existence of two algorithms, **KeyGen** and **UKeyGen**, whose distributions are indistinguishable from **TKeyGen** and **UTKeyGen**, which have threshold setting. The algorithms are constructed as follows:

KeyGen: Given *para*, generate $(a_1, a_2) \leftarrow \mathbb{Z}_p^2$ as *msk* and compute $(a_1\ P, a_2\ P) \in \mathbb{G}^2$ as *mpk*.

UKeyGen: Given $(para, mpk, ID, msk)$, generate $sk_{ID} = (a_1 H_1(ID_i), a_2 H_2(ID_i))$.

For **KeyGen**, $mpk = (a_1\ P, a_2\ P)$ is uniformly distributed as long as $msk = (a_1, a_2)$ is uniform-randomly generated. On the other hand, for **TKeyGen**, $mpk = (\sum_{j\in\mathcal{D}} a_{1,j,0}P, \sum_{j\in\mathcal{D}} a_{2,j,0}P) \in \mathbb{G}^2$ and $msk = (a_1 = \sum_{ID_i\in\mathcal{D}} a_{1,i,0}, a_2 = \sum_{ID_i\in\mathcal{D}} a_{2,i,0}) \in \mathbb{Z}_p^2)$ are uniform-randomly generated if there exists at least a single node which honestly generates a partial secret key. Thus, these distributions are indistinguishable.

Next, for **UKeyGen**, if at least t partial secret keys are collected, the output $sk_{ID} = (a_1 H_1(ID_i), a_2 H_2(ID_i))$ computed with the Lagrange interpolation is indistinguishable from that of **UTKeyGen** because their outputs are exactly identical. Furthermore, if there exists at least a single node which honestly generates a partial secret key, *mpk* and *msk* are uniform-randomly generated. Thus, these distributions are indistinguishable.

Finally, we briefly note that the algorithms described above is uncomputable for an adversary. In particular, computations for **KeyGen** and **UKeyGen** correspond to a discrete logarithm problem and CDH problems. Although we omit the details, the algebraic structures are almost identical to that of the BLS signatures [67] which are provably secure. □

REFERENCES

[1] S. D. Velagandula, N. Dhang, R. Datta, S. K. Ghosh, and S. Maroju, "Railway bridge health monitoring system using smart wireless sensor network," in *Proc. WiSec*, Jul. 2017, pp. 288–290.

[2] A. Giorgetti, M. Lucchi, E. Tavelli, M. Barla, G. Gigli, N. Casagli, M. Chiani, and D. Dardari, "A robust wireless sensor network for landslide risk analysis: System design, deployment, and field testing," *IEEE Sensors J.*, vol. 16, no. 16, pp. 6374–6386, Aug. 2016.

[3] M. Li and Y. Liu, "Underground coal mine monitoring with wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 5, no. 2, p. 10, Mar. 2009.

[4] N. Kato, "Advanced ad hoc and mesh networks technologies to facilitate specialized disaster-resilient networks," in *Proc. Trustcom*, 2014.

[5] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneç, and M. B. Yagoubi, "A survey on position-based routing protocols for flying ad hoc networks (FANETs)," *Veh. Commun.*, vol. 10, pp. 29–56, Oct. 2017.

[6] W. Zafar and B. Muhammad Khan, "Flying ad-hoc networks: Technological and social implications," *IEEE Technol. Soc. Mag.*, vol. 35, no. 2, pp. 67–74, Jun. 2016.

[7] C. Perkins, E. Belding-Royer, and S. Das, *Ad Hoc On-Demand Distance Vector (AODV) Routing*, document IETF RFC3561, 2003.

[8] D. Johnson, Y. Hu, and D. Maltz, *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*, document IETF RFC4728, 2007.

[9] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Netw.*, vol. 1, nos. 2–3, pp. 293–315, Sep. 2003.

[10] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Netw.*, vol. 13, no. 6, pp. 24–30, Nov./Dec. 1999.

[11] Y. C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless Netw.*, vol. 11, nos. 1–2, pp. 21–38, Jan. 2005.

[12] M. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *Proc. WISE*, Sep. 2002, pp. 1–10.

[13] Y.-H. Lee, H. Kim, B. Chung, J. Lee, and H. Yoon, "On-demand secure routing protocol for ad hoc network using ID based cryptosystem," in *Proc. 4th Int. Conf. Parallel Distrib. Comput., Appl. Technol. (ICPDCAT)*, Aug. 2003, pp. 211–215.

[14] J. Kim and G. Tsudik, "SRDP: Secure route discovery for dynamic source routing in MANETs," *Ad Hoc Netw.*, vol. 7, no. 6, pp. 1097–1109, Aug. 2009.

[15] G. Ács, "Secure routing in multi-hop wireless networks," Ph.D. dissertation, Dept. Telecommun., Budapest Univ. Technol. Econ., Budapest, Hungary, 2009.

[16] U. Ghosh and R. Datta, "Identity based secure AODV and TCP for mobile ad hoc networks," in *Proc. 1st Int. Conf. Wireless Technol. Humanitarian Relief (ACWR)*, Dec. 2011, pp. 339–346.

[17] U. Ghosh and R. Datta, "A novel signature scheme to secure distributed dynamic address configuration protocol in mobile ad hoc networks," in *Proc. WCNC*, Apr. 2012, pp. 2700–2705.

[18] U. Ghosh and R. Datta, "SDRP: Secure and dynamic routing protocol for mobile ad-hoc networks," *IET Netw.*, vol. 3, no. 3, pp. 235–243, Sep. 2013.

[19] U. Ghosh and R. Datta, "A Secure Addressing Scheme for Large-Scale Managed MANETs," *IEEE Trans. Netw. Service Manage.*, vol. 12, no. 3, pp. 483–495 (2015).

[20] B. May. (2009). *Ariadne: Secure On-Demand Routing in Ad-Hoc Networks—An Explanation for Dummies*. [Online]. Available: https://www.net.t-labs.tu-berlin.de/teaching/ss09/IR_seminar/talks/ariadne_may.handout.pdf

[21] K. Sanzgiri, D. LaFlamme, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "Authenticated routing for ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 3, pp. 598–610, Mar. 2005.

[22] K. Muranaka, N. Yanai, S. Okamura, and T. Fujiwara, "ISDSR: Secure DSR with ID-based sequential aggregate signature," in *Proc. ICETE*, Jul. 2016, pp. 376–387.

[23] J. Song, H. Kim, S. Lee, and H. Yoon, "Security enhancement in ad hoc network with ID-based cryptosystem," in *Proc. ICACT*, vol. 1, Feb. 2005, pp. 372–376.

[24] A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum. (2010). *Ordered Multisignatures and Identity-Based Sequential Aggregate Signatures, with Applications to Secure Routing (Extended Abstract)*. [Online]. Available: https://www.cc.gatech.edu/ aboldyre/papers/bgoy.pdf

[25] H. Kojima and N. Yanai, "Performance evaluation for the signature algorithm of ISDSR on raspberry Pi," in *Proc. CANDAR*, Nov. 2017, pp. 230–236.

[26] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[27] E. Sakhaee and A. Jamalipour, "A new stable clustering scheme for pseudo-linear highly mobile ad hoc networks," in *Proc. GLOBECOM*, Nov. 2007, pp. 1169–1173.

[28] M. Iordanakis *et al.*, "Ad-hoc routing protocol for aeronautical mobile ad-hoc networks," in *Proc. CSNDSP*, Jul. 2006, pp. 1–5.

[29] K. Muranaka, N. Yanai, S. Okamura, and T. Fujiwara, "Secure routing protocols for sensor networks: Construction with signature schemes for multiple signers," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2015, pp. 1329–1336.

[30] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 196. Berlin, Germany: Springer, 1984, pp. 47–53.

[31] F. Hidoussi, H. Toral-Cruz, D. E. Boubiche, K. Kakhtaria, A. Mihovska, and M. Voznak, "Centralized IDS based on misuse detection for cluster-based wireless sensors networks," *Wireless Pers. Commun.*, vol. 85, no. 1, pp. 207–224, Nov. 2018.

[32] L. Nishani and M. Biba, "Machine learning for intrusion detection in MANET: A state-of-the-art survey," *J. Intell. Inf. Syst.*, vol. 46, no. 2, pp. 391–407, Apr. 2016.

[33] K. Pushpalatha and M. Karthikeyan, "A generalized framework for disruption tolerant secure opportunistic routing during emergency situations using MANETs," in *Cluster Computing*. New York, NY, USA: Springer, 2018, pp. 1–9.

[34] S. D. Ubarhande, D. D. Doye, and P. S. Nalwade, "A secure path selection scheme for mobile ad hoc network," *Wireless Pers. Commun.*, vol. 97, no. 2, pp. 2087–2096, Nov. 2017.

[35] J. C. Choon and J. H. Cheon, "An identity-based signature from Gap-Diffie-Hellman groups, in *Public Key Cryptography—PKC* (Lecture Notes in Computer Science), vol. 2057. Berlin, Germany: Springer, 2003, pp. 18–30.

[36] K. Itakura and K. Nakamura,"A public-key cryptosystem suitable for digital multisignatures," *NEC Res. Develop.*, vol. 71, pp. 1–8, 1983.

[37] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme," in *Public Key Cryptography—PKC* (Lecture Notes in Computer Science), vol. 2567. Berlin, Germany: Springer, 2003, pp. 31–46.

[38] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 2656. Berlin, Germany: Springer, 2003, pp. 416–432.

[39] C. Gentry and Z. Ramzan, "Identity-based aggregate signatures," *Public Key Cryptography—PKC* (Lecture Notes in Computer Science), vol. 3958. Berlin, Germany: Springer, 2006, pp. 257–273.

[40] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proc. CCS*, Oct. 2006, pp. 390–399.

[41] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham, "Sequential aggregate signatures from trapdoor permutations," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 3027. Berlin, Germany: Springer, 2004, pp. 74–90.

[42] N. Yanai, M. Mambo, and E. Okamoto, "An ordered multisignature scheme under the CDH assumption without random oracles," in *Information Security* (Lecture Notes in Computer Science), vol. 7807. Cham, Switzerland: Springer, 2013, pp. 367–377.

[43] Y. Desmedt and Y. Frankel "Threshold cryptosystems," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 435. New York, NY, USA: Springer-Verlag, 1989, pp. 307–315.

[44] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 1070. Berlin, Germany: Springer, 1996, pp. 354–371.

[45] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 1592. Berlin, Germany: Springer, 1999, pp. 295–310.

[46] C. Park and K. Kurosawa "New EIGamal type threshold digital signature scheme," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vols. E79–A, no. 1, pp. 86–93, 1996.

[47] M. Bohio and A. Miri "Efficient identity-based security schemes for ad hoc network routing protocols," *Ad Hoc Netw.*, vol. 2, no. 3, pp. 309–317, Jul. 2004.

[48] H.-Y. Chien and R.-Y. Lin "Improved ID-based security framework for ad hoc network," *Ad Hoc Netw.*, vol. 6, no. 1, pp. 47–60, Jan. 2008.

[49] H. Deng, A. Mukherjee, and D. P. Agrawal "Threshold and identity-based key management and authentication for wireless ad hoc networks," in *Proc. ITCC*, Apr. 2004, pp. 107–111.

[50] K. Hoeper and G. Gong "Bootstrapping security in mobile ad hoc networks using identity-based schemes," in *Computer and Network Security: Security in Distributed and Networking Systems*, vol. 1. 2007, pp. 313–337.

[51] E. da Silva and L. C. P. Albini, "Towards a fully self-organized identity-based key management system for MANETs," in *Proc. WiMob*, Oct. 2013, pp. 717–723.

[52] Y. Zhang, W. Liu, W. Lou, and Y. Fang "Securing mobile ad hoc networks with certificateless public keys," *IEEE Trans. Dependable Secure Comput.*, vol. 3, no. 4, pp. 386–399, Oct./Nov. 2006.

[53] L. Buttyán and I. Vajda, "Towards provable security for ad hoc routing protocols," in *Proc. SASN*, Oct. 2004, pp. 94–105.

[54] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *Proc. CNDS* 2002, pp. 27–31.

[55] Y.-C. Hu, A. Perrig, and D. B. Johnson "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless Netw.*, vol. 11, nos. 1–2, pp. 21–38, Jan. 2005.

[56] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proc. ISCC*, Jun./Jul. 2011, pp. 850–855.

[57] E. Ghadafi, "Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 9048. Cham, Switzerland: Springer, 2015, pp. 391–409.

[58] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *Public-Key Cryptography—PKC* (Lecture Notes in Computer Science), vol. 7778. Berlin, Germany: Springer, 2013, pp. 125–142.

[59] I. Komargodski, M. Naor, and E. Yogev, "How to share a secret, infinitely," in *Proc. Theory Cryptogr. Conf.* Berlin, Germany: Springer, vol. 9986, 2016, pp. 485–514.

[60] I. Komargodski and A. Paskin-Cherniavsky, "Evolving secret sharing: Dynamic thresholds and robustness," in *Proc. Theory Cryptogr. Conf.* Cham, Switzerland: Springer, vol. 10678, 2017, pp. 379–393.

[61] M. Schulz, E. Deligeorgopoulos, M. Hollick, and F. Gringoli, "Massive reactive smartphone-based jamming using arbitrary waveforms and adaptive power control," in *Proc. WiSec*, Jul. 2017, pp. 111–121.

[62] Y. Wu, B. Wang, K. J. R. Liu, and T. C. Clancy, "Anti-jamming games in multi-channel cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 4–15, Jan. 2012.

[63] Q. Wang, K. Ren, P. Ning, and S. Hu "Jamming-resistant multiradio multichannel opportunistic spectrum access in cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 8331–8344, Oct. 2016.

[64] P. D. Thanh, H. Vu-Van, and I. Koo, "Secure multi-hop data transmission in cognitive radio networks under attack in the physical layer," *Wireless Pers. Commun.*, vol. 103, no. 2, pp. 1615–1631, Nov. 2018.

[65] B. Lantz, B. Heller, and N. Mckeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. Hotnets*, Oct. 2010, p. 19.

[66] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg, "Mininet-WiFi: Emulating software-defined wireless networks," in *Proc. CNSM*, Nov. 2015, pp. 384–389.

[67] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology—ASIACRYPT* (Lecture Notes in Computer Science), vol. 2248. Berlin, Germany: Springer, 2001, pp. 514–532.

HIDEHARU KOJIMA received the M.S. and Ph.D. degrees from Hiroshima City University, in 2001 and 2009, respectively. He is currently an Assistant Professor with the Graduate School of Information Science and Technology, Osaka University. His interests include MANET protocols and verification.

NAOTO YANAI received the B.Eng. degree from the National Institution of Academic Degrees and University Evaluation, Japan, in 2009, and the M.S.Eng. and Dr.E. degrees from the Graduate School of Systems and Information Engineering, University of Tsukuba, Japan, in 2011 and 2014, respectively. He is currently an Assistant Professor with Osaka University, Japan. His research interests include cryptography and information security.

JASON PAUL CRUZ received the B.S. degree in electronics and communications engineering and the M.S. degree in electronics engineering from the Ateneo de Manila University, Quezon, Philippines, in 2009 and 2011, respectively, and the Ph.D. degree in engineering from the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan, in 2017. He is currently a Specially Appointed Assistant Professor with Osaka University, Osaka, Japan. His current research interests include role-based access control, blockchain technology, hash functions and algorithms, privacy-preserving cryptography, and android programming.