Provisioning Protected Resource Sharing in Multi-Hop Wireless Networks

E-yong Kim[1] Hwangnam Kim[2] Kunsoo Park[1]

[1] School of Computer Science and Engineering, Seoul National University
{eykim,kpark}@theory.snu.ac.kr
[2] School of Electrical Engineering, Korea University
hnkim@korea.ac.kr

Abstract

We propose a protection framework for resource sharing to promote cooperation among nodes in multi-hop wireless networks. In the resource sharing protocol, a node claims *credits* when it relays others' packets. A node also issues *rewards* to the node which relays its packets. Rewards are used to validate the correctness of credits. In order to protect credits and rewards, we devise a secure registry scheme that supports the timed test of credit validation, and then prove that the scheme is not susceptible to various security attacks. Without any trusted authority in the operation of the framework, we make cryptographic definitions for the scheme, construct a provably secure registry scheme, and implement the timed test of credit validation with one-way chains. Finally, simulation results observed in *J-Sim* simulator corroborate that resource sharing is correctly supported and that credits and rewards are secured from selfish behaviors.

1 Introduction

In recent years, multi-hop wireless networks such as [2, 14, 27] are deployed readily since they are easy to deploy and expand. In the networks, wireless nodes are expected to relay packets for each other. Wireless nodes consume their own resource such as bandwidth and energy on relaying packets for other nodes. For that reason, *selfish nodes* may consume network resource for their traffic but do not provide their resource for others' traffic. Consequently, they make traffic concentrated on cooperative nodes and even make the network connectivity disrupted. In order to make the networks functional, cooperation among nodes is an essential requirement in multi-hop wireless networks.

Our goal is promoting cooperation among nodes by proportionally allocating resource shares according to their contribution to the network connectivity. Figure 1 shows the overview of the resource sharing protocol. The *credit* is the amount of *traffic* which a node (B) relays for other node (A). The *reward* is a tuple of $\langle \textit{issuer,credit,recipient} \rangle$ which means that the *credit* amount of *issuer* (A)'s traffic is relayed by *recipient* (B). Every node in the network carries out three operations simultaneously. First, each node accumulates its own credits when it relays traffic for others. Then it broadcasts its own credits with others' credits to neighbors. Second, each node issues rewards for neighboring nodes which relay its traffic. It also broadcasts its rewards with others' rewards to neighbors. Third, each node collects

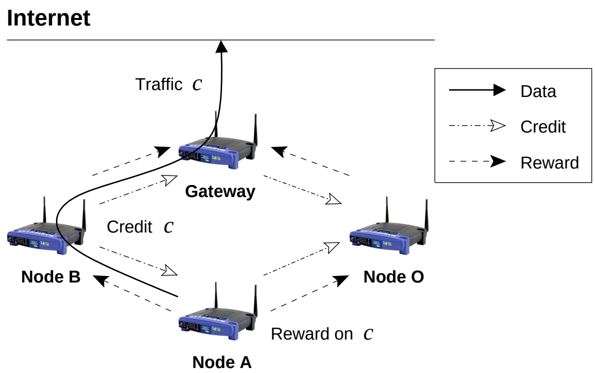


Figure 1: Overview of the resource sharing protocol in multi-hop wireless network.

other nodes' credits and rewards, and it validates credits with rewards. If credits are valid, it determines *resource share* for every node in the network according to its credits. For example, see Figure 1. Suppose that node B relays node A's traffic c. Node B accumulates its own credit c locally and broadcasts it with routing packets to neighbors, gateway and node A. Node A issues a reward on c for node B and broadcasts it to neighbors, node B and node O. Eventually, every node in the network shares the credits and rewards, and it proceeds to the test of credit validation and the resource sharing computation. In order to make this process dependable, all the information above should be protected against dropping, forgery, modification and replay attacks.

In this paper, we propose a protection framework for the resource sharing protocol in multi-hop wireless networks. We present a secure registry scheme for reward and credit which supports the *timed*[1] test of credit validation. In this scheme, the credit is the amount of relaying traffic signed with the recipient's secret key. The reward is the tuple $\langle issuer, credit, recipient\rangle$, where *credit* is concealed with the *recipient*'s public key and signed with the *issuer*'s secret key. A testing node cannot see the credit of the reward, but it can test whether the credit of the reward is equal to the credit claimed by the recipient by using bilinearity in identity-based cryptosystem. This makes our framework work distributively without interaction with any trusted authority during the operation. In addition, the test of credit validation can be *timed* by using one-way chains and delaying release of elements of the chain. It also prevents replay attack.

To show that our framework is secure against various security attacks, we present the necessary notions of security: *reward privacy*, *reward anonymity*, *reward unforgeability* and *signature unforgeability*. We prove our scheme guarantees these notions of security in the random oracle model [5]. We then implement the resource sharing protocol in *J-Sim* simulator [19] to illustrate that resource sharing is correctly performed and that credits and rewards are protected from selfish behaviors.

The rest of the paper is organized as follows. We first give preliminaries for our approach in Section 2. We describe the framework and associated security model in Section 3. We then construct a proposed scheme and prove its securities in Section 4. We show the simulation experiment to evaluate our approach in Section 5. We summarize the related work in Section 6. Finally, we conclude the paper in Section 7.

[1]The timed test means that the test comes to be effective only after a given time interval.

2 Preliminaries

2.1 Identity-Based Cryptosystem

Pairing Parameter Generator A *pairing parameter generator* is a polynomial-time randomized algorithm $\mathcal{G}$ which on input 1^k outputs a k-bit prime number q, an additive cyclic group $\mathbb{G}_1$ of prime order q, a multiplicative cyclic group $\mathbb{G}_2$ of the same order, and an efficiently computable *bilinear pairing* $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties [8]:

1. *Bilinearity*: $\forall\ g, h \in \mathbb{G}_1$, $\forall\ a, b \in \mathbb{Z}_q^*$, we have $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$.
2. *Non-degeneracy*: If g is a generator of $\mathbb{G}_1$, then $\hat{e}(g, g) \neq 1$, where 1 is the identity of $\mathbb{G}_2$.

Note that $\hat{e}(\cdot, \cdot)$ is symmetric since $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab} = \hat{e}(g^b, h^a)$. For a prime order group $\mathbb{G}_1$, we use $\mathbb{G}_1^*$ to denote $\mathbb{G}_1 \setminus \{0\}$, where 0 is the identity of $\mathbb{G}_1$.

Complexity Assumptions For the security proofs in Section 4.2, we recall the assumptions of the Bilinear Diffie-Hellman (BDH) problem and the Computational Diffie-Hellman (CDH) problem [18, 8].

Definition 2.1 (BDH assumption) The advantage of an adversary $\mathcal{A}$ in solving the BDH problem for $\mathcal{G}$ is defined as $\mathsf{Adv}_{\mathcal{G},\mathcal{A}}^{\mathrm{BDH}}(k) = \Pr[\mathcal{A}(g, g^\alpha, g^\beta, g^\gamma) = \hat{e}(g, g)^{\alpha\beta\gamma}]$ for $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle \stackrel{R}{\leftarrow} \mathcal{G}(1^k)$, $g \in_R \mathbb{G}_1^*$ and $\alpha, \beta, \gamma \in_R \mathbb{Z}_q^*$. We say that the BDH problem is hard for $\mathcal{G}$ if $\mathsf{Adv}_{\mathcal{G},\mathcal{A}}^{\mathrm{BDH}}(k)$ is negligible in k for all polynomial-time algorithms $\mathcal{A}$.

Definition 2.2 (CDH assumption) The advantage of an adversary $\mathcal{A}$ in solving the CDH problem for $\mathcal{G}$ is defined as $\mathsf{Adv}_{\mathcal{G},\mathcal{A}}^{\mathrm{CDH}}(k) = \Pr[\mathcal{A}(g, g^\alpha, g^\beta) = g^{\alpha\beta}]$ for $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle \stackrel{R}{\leftarrow} \mathcal{G}(1^k)$, $g \in_R \mathbb{G}_1^*$ and $\alpha, \beta \in_R \mathbb{Z}_q^*$. We say that the CDH problem is hard for $\mathcal{G}$ if $\mathsf{Adv}_{\mathcal{G},\mathcal{A}}^{\mathrm{CDH}}(k)$ is negligible in k for all polynomial-time algorithms $\mathcal{A}$.

2.2 One-Way Chains

We need to commit to a sequence of random coins for the *timed* test of credit validation. We use a one-way hash function to construct a *one-way chain*. Lamport first used one-way chains for one-time passwords [23]. Cheung [12], Hauser *et al.* [17] and Perrig *et al.* [29, 28] presented authentication techniques based on one-way hash chains.

A one-way chain of length ℓ is a sequence

$$r_0 \xleftarrow{F} r_1 \ \cdots\ r_{i-1} \xleftarrow{F} r_i \xleftarrow{F} r_{i+1} \ \cdots\ r_{\ell-1} \xleftarrow{F} r_\ell\ ,$$

where r_ℓ is a secret random value, $F(\cdot)$ is a one-way hash function, and $r_i = F(r_{i+1})$. Suppose that an issuer commits random values to a recipient. The issuer generates the one-way chain in the right to left direction. The issuer then releases the values to the recipient in the opposite direction. r_0 is a commitment to the entire one-way chain. To verify r_i is the ith element of the one-way chain, we check that $F^i(r_i) = r_0$ or $F^{i-j}(r_i) = r_j$ for some known r_j $(i > j)$. In our protection framework, the issuer generates a reward with r_{i+1} and releases r_i to the recipient. At this point, the recipient cannot generate r_{i+1} from r_i and hence a signature with r_{i+1}. After the issuer releasing r_{i+1}, the recipient can generate the signature which can be used to test with the reward.

3 Framework and Security Model

We formally define a credit-reward registry scheme and the associated notions of security. We then describe operations of the resource sharing protocol with credit-reward registry scheme. Finally, we show that the protocol with credit-reward registry scheme is resistant against various security attacks. We first start this section with making assumptions and defining notations used in the rest of the paper.

We make three assumptions about nodes. First, each node has a unique identity which is used both to send packets and to record traffic forwarding, and it cannot be changed arbitrarily. Such identities are not part of the current 802.11 standard, but can be provided by the forthcoming WPA2 (802.11i) security standard [25]. Each node also obtains the corresponding secret key based on its identity when the node enters into the wireless network. Second, each node can choose message sending mode, sender-identified one and sender-anonymous one. The receiver cannot determine the identity of the sender in the sender-anonymous mode. This mode can be provided for most current 802.11 hardware by scrubbing the source MAC address on packets [25]. Third, every node in the network has time synchronization with each other. All the operations are carried out at every time *interval*. Since the test of credit validation is delayed to the next time interval, we use the notion of sliding time *window*, out of which all the results are invalidated and removed permanently.

Notations To facilitate the development of the model, we refer to the network configuration in Figure 1, where we deal with only two wireless nodes A and B to employ the following notations. We use id_A to denote node A's identity and pk_A/sk_A to denote the public/secret key pair of node A. Let $\mathtt{R}_A^B(c)$ denote a reward issued by *issuer* A to *recipient* B on credit c. Let $\mathtt{S}_B^A(c)$ denote a signature of B on A's identity and credit c. We use $\mathtt{R}_A^B(c;r)$ and $\mathtt{S}_B^A(c;r)$ to denote a reward and a signature respectively with the explicit random coins r.

3.1 Definition of Scheme

We define a credit-reward registry scheme to be a tuple of polynomial-time randomized algorithms $\mathcal{RS} = \langle \mathsf{SETUP}, \mathsf{KE}, \mathsf{RWD}, \mathsf{SIG}, \mathsf{VER}, \mathsf{TEST} \rangle$, where the algorithms are as follows:

1. The setup algorithm $\mathsf{SETUP}(1^k)$ takes as input a security parameter k and outputs a system public/master key pair $\langle P, K \rangle$.
2. The key generation algorithm $\mathsf{KG}(K, id)$ takes as input the system master key K and an identity id. It outputs the corresponding secret key sk.
3. The reward generation algorithm $\mathsf{RWD}(P, sk_A, c, id_B)$ is given the system public key P, issuer's secret key sk_A, credit c and recipient's identity id_B. It generates an anonymous reward of A on c with id_B, and outputs the reward $\mathtt{R}_A^B(c)$.
4. The signature algorithm $\mathsf{SIG}(P, id_A, c, sk_B)$ is given the system public key P, an identity id_A, credit c and recipient's secret key sk_B. It constructs a signature of B on id_A and c, and outputs the signature $\mathtt{S}_B^A(c)$.
5. The verification algorithm $\mathsf{VER}(P, \mathtt{S}, id_A, c, id_B)$ takes as input the system public key P, a signature $\mathtt{S}$, an identity id_A, credit c and recipient's identity id_B. It checks the signature, and outputs `accept` if it is valid, and `reject` otherwise.
6. The test algorithm $\mathsf{TEST}(P, \mathtt{R}_A^B(c), \mathtt{S}_{B'}^{A'}(c'), id_A)$ takes as input the system public key P, a reward $\mathtt{R}_A^B(c)$, a signature $\mathtt{S}_{B'}^{A'}(c')$ and issuer's identity id_A. It tests the reward with the signature and outputs `yes` if $\langle A, c, B \rangle = \langle A', c', B' \rangle$, and `no` otherwise.

In the later part, we need to make explicit the random choices underlying algorithms. The notation $(\cdot) \stackrel{R}{\leftarrow} \mathsf{ALG}(\cdot)$ is shorthand for $r \in_R \{0,1\}^k$; $(\cdot) \leftarrow \mathsf{ALG}(\cdot; r)$, where $\mathsf{ALG}(\cdot; r)$ is run of algorithm ALG under random coins r.

3.2 Notions of Security

We consider the necessary notions of security: *reward privacy*, *reward anonymity*, *reward unforgeability* and *signature unforgeability*. These are needed for credit-reward registry schemes to protect the resource sharing protocol. The following oracles $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$ and $\mathcal{R}(\cdot,\cdot,\cdot)$ are commonly used in the security definitions.

Oracle $\mathcal{E}(id)$	Oracle $\mathcal{S}(id_A, c, id_B)$	Oracle $\mathcal{R}(id_A, c, id_B)$
$sk \leftarrow \mathsf{KG}^H(K, id)$	$sk_B \leftarrow \mathsf{KG}^H(K, id_B)$	$sk_A \leftarrow \mathsf{KG}^H(K, id_A)$
return sk	$\mathtt{S} \stackrel{R}{\leftarrow} \mathsf{SIG}^H(P, id_A, c, sk_B)$	$\mathtt{R} \stackrel{R}{\leftarrow} \mathsf{RWD}^H(P, sk_A, c, id_B)$
	return $\mathtt{S}$	**return** $\mathtt{R}$

Reward Privacy We would like to show that a reward does not reveal any information about credit without signature on it. Any adversary should not be able to distinguish a reward of one credit from a reward of another credit, where two challenge credits are of his choice. The active adversary can make adaptive chosen secret key, signature and reward queries, except that he is not allowed to issue a secret key query for the challenge identity and signature queries for any challenge credit. We now provide a formal definition.

Definition 3.1 (Privacy) Let $\mathcal{RS} = \langle \mathsf{SETUP}, \mathsf{KG}, \mathsf{RWD}, \mathsf{SIG}, \mathsf{VER}, \mathsf{TEST} \rangle$ be a credit-reward registry scheme. Let $\mathcal{A}$ be an adversary that has access to oracles $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$ and H. For a bit $b \in \{0,1\}$, we define the following experiment:

Experiment 3.1 $\mathsf{Exp}^{\text{priv-}b}_{\mathcal{RS},\mathcal{A}}(k)$

$\langle P, K \rangle \stackrel{R}{\leftarrow} \mathsf{SETUP}(1^k)$; pick random oracle H.
$\mathcal{A}$, given P, adaptively queries $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$ and H,
  and $\mathcal{A}$ outputs $(id_A^*, c_0, c_1, id_B^*)$.
$sk_A^* \leftarrow \mathsf{KG}^H(K, id_A^*)$; $\mathtt{R} \stackrel{R}{\leftarrow} \mathsf{RWD}^H(P, sk_A^*, c_b, id_B^*)$; give the challenge $\mathtt{R}$ to $\mathcal{A}$.
$\mathcal{A}$ issues additional queries and outputs a guess b'.
if $\mathcal{A}$ did not query $\mathcal{E}(id_B^*)$, $\mathcal{S}(id_A^*, c_0, id_B^*)$ and $\mathcal{S}(id_A^*, c_1, id_B^*)$
  then return b' **else return** 0

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{priv}}_{\mathcal{RS},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\text{priv-1}}_{\mathcal{RS},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{\text{priv-0}}_{\mathcal{RS},\mathcal{A}}(k) = 1]$. An $\mathcal{RS}$ is said to be *semantically secure* if $\mathsf{Adv}^{\text{priv}}_{\mathcal{RS},\mathcal{A}}(k)$ is negligible in k for all poly-time attackers $\mathcal{A}$.

Reward Anonymity Anonymity is an adaptation of [4, 1] to our scheme. We would like to show that a reward does not reveal any information about an issuer (recipient) without signature on it. Any adversary should not be able to tell which specific issuer (recipient), out of two issuers (recipients) of his choice, is the one under which the reward was created. The active adversary can make adaptive chosen secret key, signature and reward queries, except that he is not allowed to issue secret key and signature queries for any challenge identity. We now provide a formal definition.

Definition 3.2 (Anonymity) Let $\mathcal{RS} = \langle \mathsf{SETUP}, \mathsf{KG}, \mathsf{RWD}, \mathsf{SIG}, \mathsf{VER}, \mathsf{TEST} \rangle$ be a credit-reward registry scheme. Let $\mathcal{A}$ be an adversary that has access to oracles $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$ and H. For a bit $b \in \{0,1\}$, we define the following experiment:

Experiment 3.2 $\mathsf{Exp}^{\text{anon-}(b_1,b_2)}_{\mathcal{RS},\mathcal{A}}(k)$

$\langle P, K \rangle \xleftarrow{R} \mathsf{SETUP}(1^k)$; pick random oracle H.
$\mathcal{A}$, given P, adaptively queries $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$ and H,
and $\mathcal{A}$ outputs $(id^0_A, id^1_A, c^*, id^0_B, id^1_B)$.
$sk^{b_1}_A \leftarrow \mathsf{KG}^H(K, id^{b_1}_A)$; $\mathtt{R} \xleftarrow{R} \mathsf{RWD}^H(P, sk^{b_1}_A, c^*, id^{b_2}_B)$; give the challenge $\mathtt{R}$ to $\mathcal{A}$.
$\mathcal{A}$ issues additional queries and outputs a pair of guesses (b'_1, b'_2).
if $\mathcal{A}$ did not query $\mathcal{E}(id^0_B)$, $\mathcal{E}(id^1_B)$, $\mathcal{S}(id^0_A, c^*, id^0_B)$, $\mathcal{S}(id^0_A, c^*, id^1_B)$, $\mathcal{S}(id^1_A, c^*, id^0_B)$ and $\mathcal{S}(id^1_A, c^*, id^1_B)$ **then return** (b'_1, b'_2) **else return** $(0,0)$

The advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}^{\text{anon}}_{\mathcal{RS},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{\text{anon-}(1,1)}_{\mathcal{RS},\mathcal{A}}(k) = (1,1)] - \Pr[\mathsf{Exp}^{\text{anon-}(1,0)}_{\mathcal{RS},\mathcal{A}}(k) = (1,1)] - \Pr[\mathsf{Exp}^{\text{anon-}(0,1)}_{\mathcal{RS},\mathcal{A}}(k) = (1,1)] - \Pr[\mathsf{Exp}^{\text{anon-}(0,0)}_{\mathcal{RS},\mathcal{A}}(k) = (1,1)]$. An $\mathcal{RS}$ is said to be *anonymous* if $\mathsf{Adv}^{\text{anon}}_{\mathcal{RS},\mathcal{A}}(k)$ is negligible in k for all poly-time attackers $\mathcal{A}$.

Reward Unforgeability We would like to show that a reward cannot be constructed without secret key. Any forger should not be able to construct any new reward on forged issuer's identity, credit and recipient's identity of his choice, for which TEST returns $\mathtt{yes}$ with the corresponding signature and the forged issuer's identity. The active forger can make adaptive chosen secret key, signature and reward queries. The only restriction is that he has not obtained the secret key for the forged issuer's identity and the forged reward for the forged issuer's identity, credit and recipient's identity. We now provide a formal definition.

Definition 3.3 (Unforgeability) Let $\mathcal{RS} = \langle \mathsf{SETUP}, \mathsf{KG}, \mathsf{RWD}, \mathsf{SIG}, \mathsf{VER}, \mathsf{TEST} \rangle$ be a credit-reward registry scheme. Let $\mathcal{F}$ be an adversary that has access to oracles $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$ and H. We define the following experiment:

Experiment 3.3 $\mathsf{Exp}^{\text{euf}}_{\mathcal{RS},\mathcal{F}}(k)$

$\langle P, K \rangle \xleftarrow{R} \mathsf{SETUP}(1^k)$; pick random oracle H.
$\mathcal{F}$, given P, adaptively queries $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$ and H,
and $\mathcal{F}$ outputs $(\mathtt{R}^*, id^*_A, c^*, id^*_B)$.
$sk^*_B \leftarrow \mathsf{KG}^H(K, id^*_B)$; $\mathtt{S}^* \xleftarrow{R} \mathsf{SIG}^H(P, id^*_A, c^*, sk^*_B)$
if $\mathsf{TEST}^H(P, \mathtt{R}^*, \mathtt{S}^*, id^*_A) = \mathtt{yes}$ and $\mathcal{F}$ did not query $\mathcal{E}(id^*_A)$ and $\mathcal{F}$ did not obtain $\mathtt{R}^*$ by a $\mathcal{R}(id^*_A, c^*, id^*_B)$ query **then return** 1 **else return** 0

The advantage of $\mathcal{F}$ is defined as $\mathsf{Adv}^{\text{euf}}_{\mathcal{RS},\mathcal{F}}(k) = \Pr[\mathsf{Exp}^{\text{euf}}_{\mathcal{RS},\mathcal{F}}(k) = 1]$. We say that an $\mathcal{RS}$ is *existentially unforgeable* if $\mathsf{Adv}^{\text{euf}}_{\mathcal{RS},\mathcal{F}}(k)$ is negligible in k for all poly-time forgers $\mathcal{F}$.

Signature (Strong Existential) Unforgeability The algorithms SIG and VER with SETUP and KG in credit-reward registry scheme should be secure signature scheme independently. We consider the notion of strong existential unforgeability previously considered in [3, 6, 24].

Definition 3.4 Let $\mathcal{IBS} = \langle \mathsf{SETUP}, \mathsf{KG}, \mathsf{SIG}, \mathsf{VER} \rangle$ be an identity-based signature scheme. Let $\mathcal{F}$ be an adversary that has access to oracles $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, and H. We define the following experiment:

Experiment 3.4 $\mathsf{Exp}^{\text{euf-cma}}_{\mathcal{IBS},\mathcal{F}}(k)$

$\langle P, K \rangle \overset{R}{\leftarrow} \mathsf{SETUP}(1^k)$; pick random oracle H.
$\mathcal{F}$, given P, adaptively queries $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$ and H, and $\mathcal{F}$ outputs $(\mathtt{S}^*, id^*_A, c^*, id^*_B)$.
if $\mathsf{VER}^H(P, \mathtt{S}^*, id^*_A, c^*, id^*_B) = \mathtt{accept}$ and $\mathcal{F}$ did not query $\mathcal{E}(id^*_B)$ and $\mathcal{F}$ did not obtain $\mathtt{S}^*$ by a $\mathcal{S}(id^*_A, c^*, id^*_B)$ query **then return** 1 **else return** 0

The *EUF-CMA-advantage* of $\mathcal{F}$ is defined as $\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{IBS},\mathcal{F}}(k) = \Pr[\mathsf{Exp}^{\text{euf-cma}}_{\mathcal{IBS},\mathcal{F}}(k) = 1]$. We say that an $\mathcal{IBS}$ is *(strongly) existentially unforgeable under chosen-message attacks* if $\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{IBS},\mathcal{F}}(k)$ is negligible in k for all poly-time forgers $\mathcal{F}$.

3.3 Operations of Protocol

We describe operations of the resource sharing protocol with credit-reward registry scheme. We also explain the timed test of credit validation supported in our framework.

A		B		T
RWD $(P, sk_A, c, id_B; r)$	$\xrightarrow{\mathtt{R}^B_A(c;r)}$	Forward $\mathtt{R}^B_A(c;r)$	$\xrightarrow{\mathtt{R}^B_A(c;r)}$	Save $\mathtt{R}^B_A(c;r)$
	$\xrightarrow{r}$	$\mathsf{SIG}(P, id_A, c, sk_B; r)$	$\xrightarrow[\mathtt{S}^A_B(c;r)]{\langle id_A, c\rangle}$	$\mathsf{VER}(P, \mathtt{S}^A_B(c;r), id_A, c, id_B)$
				$\downarrow$ `accept`
				$\mathsf{TEST}(P, \mathtt{R}^B_A(c;r), \mathtt{S}^A_B(c;r), id_A)$
				$\downarrow$ `yes`
*For simple description, we omit subscripts.				Compute *resource share*

Figure 2: Operations of the resource sharing protocol with credit-reward registry scheme. Node A generates an anonymous reward $\mathtt{R}^B_A(c;r)$ for relaying its traffic and broadcasts it to neighboring nodes. Neighbors (including B) rebroadcast the anonymous reward. Later, node B broadcasts $\langle id_A, c\rangle$ with the signature $\mathtt{S}^A_B(c;r)$ on it. Eventually, every node in the network shares others' credits and rewards. Node T verifies the received signature with the credit, and tests anonymous rewards with the accepted signature. Finally, if there is only one `yes` answer, T proceeds to the computation of *resource share* according to the credit c.

We suppose that every node in the network agrees with the interval duration, start time and interval index i. In interval i, every node broadcasts rewards and credits of interval i, and every node validates credits of interval $i-1$ with rewards of interval $i-1$. Suppose node B relays node A's traffic, the amount of which is c_i (See Figure 2).

1. **Reward Generation and Broadcast**: A picks a random number r_{i+1} and generates an anonymous reward $\mathtt{R}_A^B(c_i; r_{i+1})$ by running algorithm RWD with B's identity and the credit c_i under the secret key sk_A and the random number r_{i+1}. Finally, A broadcasts $\mathtt{R}_A^B(c_i; r_{i+1})$ to neighboring nodes, releases previously saved r_i to B and saves r_{i+1} in the local disk.
2. **Credit Signing and Broadcast**: B records his current credit record $\langle id_A, c_i \rangle$ in the local disk and reads the previous credit record $\langle id_A, c_{i-1} \rangle$. After receiving the random number r_i from A, B generates the signature $\mathtt{S}_B^A(c_{i-1}; r_i)$ by running algorithm SIG with the credit record $\langle id_A, c_{i-1} \rangle$ under the random number r_i. B then broadcasts his current credit record $\langle id_A, c_i \rangle$ with the signature $\mathtt{S}_B^A(c_{i-1}; r_i)$ to neighboring nodes.
3. **Credit Validation Test**: Eventually, T receives $\mathtt{R}_A^B(c_i; r_{i+1})$ and $\langle id_A, c_i \rangle, \mathtt{S}_B^A(c_{i-1}; r_i)$. T saves $\mathtt{R}_A^B(c_i; r_{i+1})$ and $\langle id_A, c_i \rangle$ in the local disk. T verifies the received signature $\mathtt{S}_B^A(c_{i-1}; r_i)$ by running algorithm VER with the previously saved credit record $\langle id_A, c_{i-1} \rangle$ and B's identity. If the signature is accepted, T tests previously saved rewards by running algorithm TEST with the signature $\mathtt{S}_B^A(c_{i-1}; r_i)$ and A's identity. If $\mathtt{R}_A^B(c_{i-1}; r_i)$ exists in the local disk, it makes `yes` output. Note that the test for credit c_i is delayed to the next interval $i + 1$.
4. **Resource Share Computation**: T does not validate credit c_i of the current interval, but c_{i-1} of the previous interval. If T discovers cheat on the previous credit c_{i-1}, it will give disadvantage to selfish node in the resource share computation of the current interval i.

3.4 Thwarting Attacks

We can address the following attacks when we operate the resource sharing protocol with credit-reward registry scheme which guarantees the notions of security in Section 3.2.

Credit Attacks The *credit attack* is an attack to forge node's own credit or others. The first attack is gaining more resource share using *fake credit* without relaying traffic for others. The concept of reward is devised to directly address such a credit forgery. The forger should generate the corresponding reward to pass TEST with his fake credit. The *reward unforgeability* prevents any one from generating other issuer's reward.

The second attack is obtaining more resource share by decreasing other credits during relaying, so as to get more resource share than the original one. The attacker should be able to generate the corresponding signature on the modified credit. The *signature unforgeability* makes such an attack impossible. Even if the attacker drops other credits, every node in the network eventually receives them by the rich connectivity of multi-hop wireless networks.

Reward Attacks The *reward attack* is that the attacker modifies or drops other nodes' rewards in order to obtain more resource share by making their TEST failed. The attacker should be able to distinguish his reward from others in order to mount this attack. However, *reward privacy* does not allow attackers to retrieve credit information, and *reward anonymity* does not allow attackers to retrieve issuer and recipient information from reward.

A malicious node would try to issue *fake reward* to the neighbors who relay his packets. However, the malicious one also gets disadvantage since it will decrease resource share for his neighbors and consequently his share also.

Replay Attack The *replay attack* means reusing previously-issued reward without relaying packets for others after once obtaining reward. The attacker should know which reward is issued for him. Again, *reward anonymity* does not allow attackers to retrieve issuer or recipient information from reward. Even if the attacker knows the reward for him by some way, the effect of the attack is very limited since all rewards are invalidated out of time *window*.

4 Credit-Reward Registry Scheme

In this section, we give a construction of credit-reward registry scheme. We then prove that the scheme guarantees the notions of security of Section 3.2 in the random oracle model. Finally, we explain how the scheme supports the timed test of credit validation using one-way chains.

4.1 Construction

Our construction of credit-reward registry scheme is based on the BDH and CDH problems (Section 2.1). Basically, the scheme is constructed in identity-based cryptosystem. The novel technique in this construction is that we use signature as a kind of trapdoor for the test of credit validation, in contrast to [7, 31, 1, 21]. This enables our scheme to operate without interaction with any trusted authority. We construct a signature scheme, which is modified from [11] and [24], so as to be suitable to our test mechanism. The construction is given in Scheme 4.1.

If $r = r'$, the credit-reward registry scheme 4.1 is computationally consistent [1] since

$$\begin{aligned} \hat{e}(g^s, pk_B^{H_2(id_A,c,u)}) &= \hat{e}(g, sk_B^{H_2(id_A,c,u)}) = \hat{e}(g, v) \ , \\ \hat{e}((g^s)^r, pk_B^{H_2(id_A,c,x)}) &= \hat{e}(g^r, sk_B^{H_2(id_A,c,u)}) = \hat{e}(x, v) \text{ and} \\ \hat{e}(g^s, pk_A^{H_3(\hat{e}(x,v))}) &= \hat{e}(g, sk_A^{H_3(\hat{e}(x,v))}) = \hat{e}(g, y) \ . \end{aligned}$$

4.2 Security Analysis

We analyze the securities for the scheme 4.1 under the BDH and CDH assumptions in the security model of Section 3.2. The proofs are given in Appendix A.

Theorem 4.1 (Privacy) *If the BDH assumption holds for $\mathcal{G}$, the credit-reward registry scheme 4.1 is semantically secure in the random oracle model.*

Theorem 4.2 (Anonymity) *If the BDH assumption holds for $\mathcal{G}$, the credit-reward registry scheme 4.1 is anonymous in the random oracle model.*

Theorem 4.3 (Unforgeability) *If the CDH assumption holds for $\mathcal{G}$, the credit-reward registry scheme 4.1 is existentially unforgeable in the random oracle model.*

Theorem 4.4 *If the CDH assumption holds for $\mathcal{G}$, the identity-based signature scheme $\langle \mathsf{SETUP}, \mathsf{KG}, \mathsf{SIG}, \mathsf{VER} \rangle$ in the scheme 4.1 is existentially unforgeable under chosen-message attacks in the random oracle model.*

Scheme 4.1 Credit-reward registry scheme

Generate pairing parameters $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle \leftarrow \mathcal{G}(1^k)$. Choose collision resistant hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$.

$\mathsf{SETUP}(1^k)$: takes a security parameter k as input.

1. Choose a random generator $g \in \mathbb{G}_1^*$ and a random $s \in \mathbb{Z}_q^*$ and compute $g^s \in \mathbb{G}_1$.
2. Output the system public key $P \leftarrow \langle g, g^s\rangle \in \mathbb{G}_1 \times \mathbb{G}_1$ and the system master key $K \leftarrow s \in \mathbb{Z}_q^*$.

$\mathsf{KG}(K, id)$: takes the system master key $K = s \in \mathbb{Z}_q^*$ and an identity $id \in \{0,1\}^*$ as input.

1. Compute the public key $pk_{id} \leftarrow H_1(id) \in \mathbb{G}_1$.
2. Output the secret key $sk_{id} \leftarrow pk_{id}^s \in \mathbb{G}_1$, where s is the master key.

$\mathsf{RWD}(P, sk_A, c, id_B; r)$: takes as input the system public key P, credit $c \in \{0,1\}^*$ and an identity $id_B \in \{0,1\}^*$ with a secret key $sk_A \in \mathbb{G}_1$ and a random number $r \in \mathbb{Z}_q^*$.

1. Compute $x \leftarrow g^r \in \mathbb{G}_1$, $h_2 \leftarrow H_2(id_A, c, x) \in \mathbb{Z}_q^*$ and $\omega \leftarrow \hat{e}((g^s)^r, pk_B^{h_2}) \in \mathbb{G}_2$, where $pk_B \leftarrow H_1(id_B) \in \mathbb{G}_1$. Finally, compute $h_3 \leftarrow H_3(\omega)$ and $y \leftarrow sk_A^{h_3} \in \mathbb{G}_1$.
2. Output the anonymous reward $\mathtt{R}_A^B(c) \leftarrow \langle x, y\rangle \in \mathbb{G}_1 \times \mathbb{G}_1$.

$\mathsf{SIG}(P, id_A, c, sk_B; r')$: takes as input the system public key P, an identity $id_A \in \{0,1\}^*$ and credit $c \in \{0,1\}^*$ with a secret key $sk_B \in \mathbb{G}_1$ and a random number $r' \in \mathbb{Z}_q^*$.

1. Compute $u \leftarrow g^{r'} \in \mathbb{G}_1$, $h_2 \leftarrow H_2(id_A, c, u) \in \mathbb{Z}_q^*$ and $v \leftarrow sk_B^{h_2} \in \mathbb{G}_1$.
2. Output the signature $\mathtt{S}_B^A(c) \leftarrow \langle u, v\rangle \in \mathbb{G}_1 \times \mathbb{G}_1$.

$\mathsf{VER}(P, \mathtt{S}_B^A(c), id_A, c, id_B)$: takes as input the system public key P, a signature $\mathtt{S}_B^A(c) = \langle u, v\rangle \in \mathbb{G}_1 \times \mathbb{G}_1$, an identity $id_A \in \{0,1\}^*$ and credit $c \in \{0,1\}^*$ with an identity $id_B \in \{0,1\}^*$.

1. Compute $pk_B \leftarrow H_1(id_B) \in \mathbb{G}_1$ and $h_2 \leftarrow H_2(id_A, c, u) \in \mathbb{Z}_q^*$.
2. Output `accept` if $\hat{e}(g, v) \stackrel{?}{=} \hat{e}(g^s, pk_B^{h_2})$, output `reject` otherwise.

$\mathsf{TEST}(P, \mathtt{R}_A^B(c), \mathtt{S}_B^A(c), id_A)$: takes as input the system public key P, a reward $\mathtt{R}_A^B(c) = \langle x, y\rangle \in \mathbb{G}_1 \times \mathbb{G}_1$ and a signature $\mathtt{S}_B^A(c) = \langle u, v\rangle \in \mathbb{G}_1 \times \mathbb{G}_1$ with an identity $id_A \in \{0,1\}^*$.

1. Compute $pk_A \leftarrow H_1(id_A) \in \mathbb{G}_1$, $\omega \leftarrow \hat{e}(x, v) \in \mathbb{G}_2$ and $h_3 \leftarrow H_3(\omega) \in \mathbb{Z}_q^*$.
2. Output `yes` if $\hat{e}(g, y) \stackrel{?}{=} \hat{e}(g^s, pk_A^{h_3})$, output `no` otherwise.

4.3 Timed Test of Credit Validation

In our base scheme 4.1, if $r = r'$ in the same interval, a relaying node can test incoming rewards with its own signature for its credit and distinguish rewards issued for it from others. In case of selfish nodes, they drop rewards for other nodes and only forward rewards issued for them.

interval	credit	random coins ($A \rightarrow B$)	reward	signature
$i-1$	c_{i-1}	r_{i-1}	$\mathtt{R}_A^B(c_{i-1}; r_i)$	$\mathtt{S}_B^A(c_{i-2}; r_{i-1})$
i	c_i	r_i	$\mathtt{R}_A^B(c_i; r_{i+1})$	$\mathtt{S}_B^A(c_{i-1}; r_i)$
$i+1$	c_{i+1}	r_{i+1}	$\mathtt{R}_A^B(c_{i+1}; r_{i+2})$	$\mathtt{S}_B^A(c_i; r_{i+1})$

For preventing such a malicious attack, we propose the *timed test of credit validation* which completes our protection framework. We use one-way chains in Section 2.2 for that purpose. Suppose that node B relays node A's traffic in Figure 3. For the convenience, we

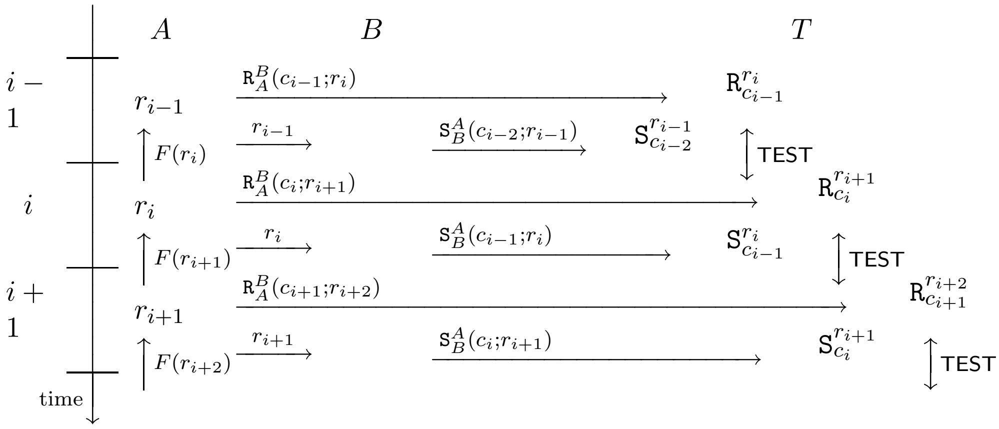


Figure 3: Overview of the timed test of credit validation. The issuer A publishes rewards to the recipient B. The reward issued at interval $i-1$ can be tested with the signature of interval i, the reward issued at interval i can be tested with the signature of interval $i+1$, and so on.

use $\mathtt{R}_{c_i}^{r_j}$ to denote $\mathtt{R}_A^B(c_i;r_j)$ and $\mathtt{S}_{c_i}^{r_j}$ to denote $\mathtt{S}_B^A(c_i,r_j)$.

1. **Issuer Setup**: A determines the length ℓ of the one-way chain $r_0, r_1, \ldots, r_\ell$, and this length limits the maximum interval before a new one-way chain must be created. Using a pseudo-random function f, A constructs the one-way function F: $F(k) = f_k(0)$. The remainder of the chain is computed recursively using $r_i = F(r_{i+1})$.
2. **Reward Broadcast and Random Coin Release**: In interval i, A constructs a reward $\mathtt{R}_{c_i}^{r_{i+1}}$ with credit c_i and the next random number r_{i+1}. A broadcasts $\mathtt{R}_{c_i}^{r_{i+1}}$ to neighbors anonymously and releases the random coin r_i to B.
3. **Signature Broadcast**: After receiving the random coin r_i, B checks that $F(r_i) = r_{i-1}$. If r_i is correct, B generates a signature $\mathtt{S}_{c_{i-1}}^{r_i}$ with the credit c_{i-1} of the previous interval and the received random coin r_i. B also broadcasts $\mathtt{S}_{c_{i-1}}^{r_i}$ to neighbors. Note that B cannot test anonymous reward $\mathtt{R}_{c_i}^{r_{i+1}}$ with its signature since it cannot generate r_{i+1} from r_i and hence the signature with r_{i+1} in the current interval.
4. **Timed Credit Validation Test**: Eventually, every node in the network shares $\mathtt{R}_{c_i}^{r_{i+1}}$ and $\mathtt{S}_{c_{i-1}}^{r_i}$. T saves $\mathtt{R}_{c_i}^{r_{i+1}}$ in the local disk. Then T tests $\mathtt{S}_{c_{i-1}}^{r_i}$ with rewards which were saved in interval $i-1$. If there is $\mathtt{R}_{c_{i-1}}^{r_i}$ among the rewards, it makes `yes` output.

We here set the time window size as 2 which can be lengthened by delaying release time of random coins. When there is no reward to be issued in some interval, the random coin in that interval also need not to be released. We still can check the next random coin using the previous one by one-way chains.

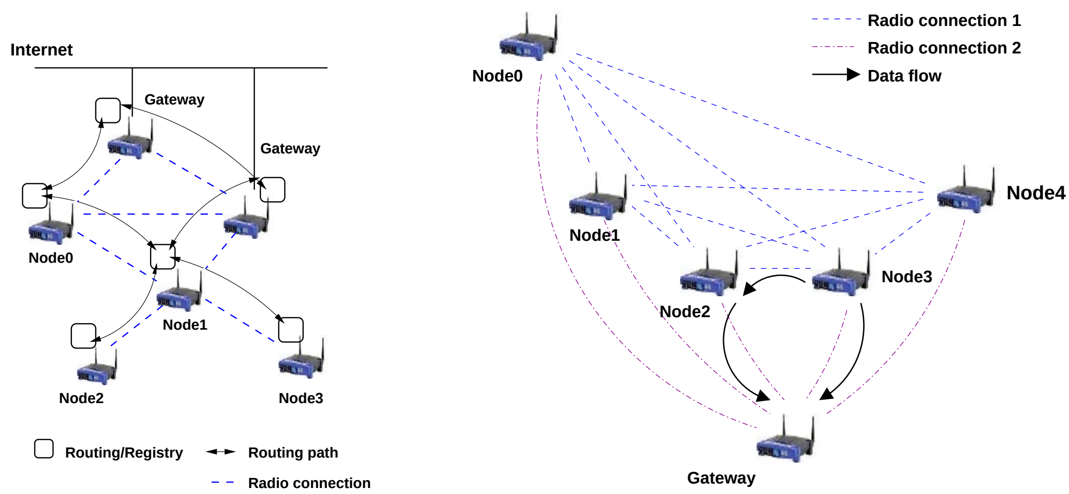
5 Experiment

The *Protected Resource Sharing (PRS)* protocol is the resource sharing protocol with the credit-reward registry scheme. In the *PRS* in Figure 4(a), each node piggybacks the reports of *rewards* and *credits* in routing packets. Ad-hoc routing protocol assumed in the *PRS* protocol is a link-state, proactive one since the *PRS* needs periodical reports and global spread of the reports.

We have implemented the proposed *PRS* of securing *rewards* in *J-Sim* V 1.3 [19], and conducted a performance study to evaluate its correctness and properties in IEEE 802.11b-operated multi-hop wireless networks. The fact that *J-Sim* contains a complete wireless extension with the IEEE 802.11 link layer, IEEE 802.11, and wireless physical layer easily facilitates the simulation study. The evaluation is made to see whether or not resource sharing per node is proportional to the contribution it made, how the overall system is affected by selfish behaviors, and how the *PRS* protects the cooperative nodes from selfish behaviors.

We use in this evaluation the network configuration presented in Figure 4(b). In Figure 4(b), each wireless node can directly reach the gateway to the Internet and also provides the other nodes with the indirect path to the gateway. For example, *node3* can communicate with *gateway* directly (presented by right arrow) or indirectly via *node2* (presented two left arrows) in the figure. The channel in the lower part of the figure, used to connect the nodes and the gateway, do not interfere with the channel in the upper part of the figure, used to connect among nodes, which can be enabled when each node is equipped multiple interfaces and uses orthogonal channels for each interface or when each channel uses one channel in time-divisional way. However, within the same radio channel, all the nodes, inclusive of the gateway, compete for the channel.

We assume that each flow $i, 0 \leq i \leq 4$, generates their traffic at 200, 150, 100, 50, and 250, respectively, and that each route alternates the direct path to indirect path, or vice versa. The traffic rates do not saturate the wireless medium because we would like to clearly see the effect of *PRS*, but we have the results for the other case even though we do not present them due to space constraint. We assume that node4 decides to be selfish during the period of [200, 300]. Additionally, we schedule each node to report *credits* and *rewards* to every node at every *interval* of five (5) seconds, and also instruct each node to allocate at most 20% of its network bandwidth for relaying other nodes. We use default values defined in *J-Sim* for IEEE 802.11-related system parameters but we use 1 Mb/s for the channel rate otherwise stated. All the simulations are conducted on Linux 2.6.11 running on a Pentium 4/3.20 GHz PC with 512 MBytes of main memory and 512 MBytes of swap memory.



(a) The *PRS* protocol. (b) Network configuration.

Figure 4: Simulation configuration for the *PRS* protocol.

We employ the following simulation scenarios for the experiments where i is a natural number such that $i \geq 0$.

- All the flows, generated at each node, are assumed to have the same initial *credits*, so that each node gets the minimum share even if it does not have the chance to forward other traffic;
- Flow 0, generated at node0, serves other flows in the intervals of $[100\cdot(i+1), 100\cdot(i+2)]$; It uses its direct path of {node0, gateway} in the intervals of $[100\cdot(i+1), 100\cdot(i+2)]$ and $[200\cdot i, 200\cdot i+20]$, but it changes its path to its indirect path of {node0, node4, gateway} in the periods of $[200\cdot i+20, 200\cdot i+100]$.
- Flow 1 changes its route from the path of {node1, gateway} to the path of {node1, node4, gateway} at the time instant $(i\cdot 200+40)$, and reverts to its direct path at $(i\cdot 200+100)$. Additionally, it changes from its direct path to another indirect path of {node1, node0, gateway} at $(i\cdot 100+20)$, and again reverts to its original path at $(i\cdot 100+100)$;
- Flow 2 uses its direct path in the intervals of $[i\cdot 200, i\cdot 200+60]$, $[i\cdot 100, i\cdot 100+40]$, but it employs the indirect path of {node2, node4, gateway} in the periods of $[i\cdot 200+60, i\cdot 200+100]$ while it does another indirect path of {node2, node0, gateway} in $[i\cdot 100+40, i\cdot 100+100]$;
- Flow 3 employs the relaying services at node4 in the interval of $[i\cdot 200+60, i\cdot 200+100]$, and those services at node 0 in $[i\cdot 100+60, i\cdot 100+100]$. Except the intervals, it uses its direct path of {node3, gateway};
- Node4 directly sends Flow 4 to the gateway at the intervals of $[i\cdot 200+0, i\cdot 200+100]$ and $[i\cdot 100, i\cdot 100+80]$ while it uses the relaying service of node 0 in the interval of $[i\cdot 100+80, i\cdot 100+100]$, and additionally forwards other flows every intervals of $[i\cdot 200, i\cdot 200+100]$.

Figure 5 presents how the *PRS* protects and purnishes selfish behaviors temporally presented to the networks. We have the following observations from the results. First, when each flow changes its route from direct path to indirect path, it shares with other flows the available bandwidth at the relaying node, node0 or node4. Second, the accumulated *credits* definitely guarantee the more resource share. For example, flow 4 obtains more resource share at the intervals of [180,200] and [480,500], and flow 0 enjoys more bandwidth during the intervals of [420,500] and [520, 600]. Last, the selfish behavior of node4 during [200,300] cannot give more bandwidth for it during [300,400]. As explained, the each node invalidates the forged *credits*. After validation, each node computes the resource shares for the nodes according to their *credits*, and so the selfish node, node4, gets the least amount of resource share. Note, however, that the effect of transient selfish behavior gradually disappear as time goes by after it decides to return to the cooperation.

6 Related Work

We have many approaches to address *free-riding* problem in various network field, but there exist few approaches to directly deal with how to protect the cooperation.

Approaches on Wireless Ad-hoc Networks Marti *et al.* [26] proposed to identify misbehaving nodes by overhearing transmissions, called watchdog, and then to assist routing protocols so as to avoid the nodes, and thus they improved the throughput in ad-hoc networks. Nuglet [10] stimulates wireless nodes to forward other traffic by saving their forwarding

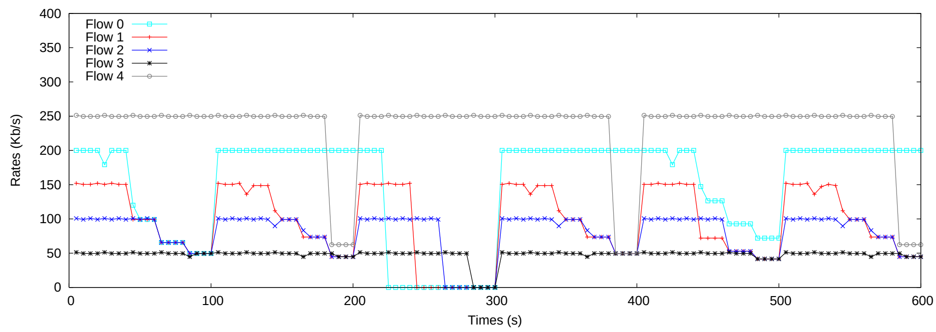


Figure 5: Resource sharing temporally disturbed by selfish behavior in the *PRS* protocol.

information in a tamper-resistant hardware module to be used to send their traffic in future. Sprite [32] accumulates credits for the nodes when they forward other traffic, and uses them later to send their packets. Each node reports credits to a trusted central authority but not in a secure way. CONFIDANT [9] not only detects selfish behaviors with the watchdog technique as in [26], but also isolates the nodes from the cooperation. Catch [25] improves the watchdog used in [26, 9] by using anonymous messages, and detects free-riders with it to isolate them from the rest of the network. In game theoretic approaches, such as in [30, 15], the network system converges to the rational and optimal operating point in the course of collaboration. Note that any credit- or incentive-based approach can be integrated with our protection mechanism to protect credits or incentives.

Approaches on P2P Systems Lai *et al.* [22] employed incentive techniques in the framework of prisoner's dilemma to count all contributions and consumption to discourage the free-riding problem. Feldman *et al.* [16] classified each user with his intrinsic characteristics *type*, and allowed him to choose whether to contribute or to free-ride on the system according to the current *cost* and his type. Jun and Ahamad [20] used a prisoner's dilemma to propose a new incentive scheme for BitTorrent.

7 Conclusion

We propose a protection framework which protects the resource sharing protocol from selfish behaviors in multi-hop wireless networks. Each node accumulates *credits* for relaying traffic for others and issues *rewards* to relay nodes for its successfully transmitted traffic. Each node broadcasts *credits* and *rewards* to the network, anonymously in case of rewards. Each node collects rewards and credits, validates the credits with rewards, then computes resource share for each node. We formally define a credit-reward registry scheme and associate notions of security: reward privacy, anonymity, unforgeability and signature unforgeability, for protecting the resource sharing protocol. We then construct a credit-reward registry scheme and prove that it guarantees the above notions of security in the random oracle model. Using one-way chains, we support the *timed test of credit validation* in our framework. Finally, we implemented the resource sharing protocol in *J-Sim* simulator and illustrate that the resource sharing is correctly performed under the protection of our credit-reward registry scheme.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer-Verlag, 2005.

[2] D. Aguayo, J. Bicket, S. Biswas, D. S. J. De Couto, and R. Morris. MIT Roofnet Implementation, Aug 2003. `http://pdos.csail.mit.edu/roofnet/design/`.

[3] J. H. An, Y. Dodis, and T. Rabin. On the Security of Joint Signature and Encryption. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer-Verlag, 2002.

[4] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes on Computer Science*, pages 566–582. Springer-Verlag, 2001.

[5] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS)*, pages 62–73, Nov 1993.

[6] D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public-Key Encryption with keyword Search. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer-Verlag, 2004.

[8] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, 32(3):586–615, Mar 2003.

[9] S. Buchegger and J.-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks). In *Proceedings of ACM MobiHoc '02*, pages 226–236, Jun 2002.

[10] L. Buttyán and J.-P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *Mobile Networks and Applications*, 8(5):579–592, Oct 2003.

[11] J. C. Cha and J. H. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. In *Proceedings of PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, 2003.

[12] S. Cheung. An Efficient Message Authentication Scheme for Link State Routing. In *Proceedings of the 13th Annual Computer Security Applications Conference (ACSAC)*, pages 90–98, Dec 1997.

[13] J.-S. Coron. On the Exact Security of Full Domain Hash. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, 2000.

[14] Champaign-Urbana Community Wireless Network. `http://www.cuwireless.net/`.

[15] Z. Fang and B. Bensaou. Fair Bandwidth Sharing Algorithms based on Game Theory Frameworks for Wireless Ad-hoc Networks. In *Proceedings of IEEE INFOCOM*, volume 2, pages 1284–1295, Mar 2004.

[16] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proceedings of the Fifth ACM Conference on Electronic Commerce*, May 2004.

[17] R. Hauser, A. Przygienda, and G. Tsudik. Reducing the Cost of Security in Link State Routing. In *Proceedings of Symposium on Network and Distributed Systems Security (NDSS)*, pages 93–99, Feb 1997.

[18] A. Joux. The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In *Proceedings of the Fifth Algorithm Number Theory Symposium (ANTS-V)*, volume 2369 of *Lecture Notes on Computer Science*, pages 20–32. Springer-Verlag, 2002.

[19] J-Sim. `http://www.j-sim.org/`.

[20] S. Jun and M. Ahamad. Incentives in BitTorrent Induce Free Riding. In *Proceeding of the 2005 ACM SIGCOMM Workshop on the Economics of Peer-to-Peer Systems*, pages 116–121, Aug 2005.

[21] E. Kim, L. Xiao, K. Nahrstedt, and K. Park. Identity-Based Registry for Secure Interdomain Routing. In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, pages 321–331, Mar 2006.

[22] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for Cooperation in Peer-to-Peer Networks. In *Proceedings of Workshop on Economics of Peer-to-Peer Systems*, Jun 2003.

[23] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, Nov 1981.

[24] B. Libert and J.-J. Quisquater. The Exact Security of an Identity Based Signature and its Applications. Cryptology ePrint Archive, Report 2004/102, 2004. Available at `http://eprint.iacr.org/2004/102`.

[25] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining Cooperation in Multihop Wireless Networks. In *Proceedings of 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[26] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of ACM MobiCom '00*, pages 255–265, Aug 2000.

[27] Microsoft Networking Research Group. Self-Organizing Neighborhood Wireless Mesh Networks. `http://research.microsoft.com/mesh/`.

[28] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and Secure Source Authentication for Multicast. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, pages 35–56, Feb 2001.

[29] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 56–73, May 2000.

[30] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao. Cooperation in Wireless Ad Hoc Networks. In *Proceedings of IEEE INFOCOM*, volume 2, pages 808–817, Mar 2003.

[31] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters. Building an Encrypted and Searchable Audit Log. In *Proceedings of the 11th Network and Distributed System Security Symposium (NDSS)*, pages 205–214, Feb 2004.

[32] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1987–1997, Mar 2003.

A Proofs of Security

The proofs of four security theorems use common simulations of the oracles. We first describe the simulations in Figure 6. In our proofs, we use Coron's proof technique [13]. Note that we use the list L_r between the oracles $\mathcal{S}(\cdot,\cdot,\cdot)$ and $\mathcal{R}(\cdot,\cdot,\cdot)$ for simulating the timed test of credit validation. We record random numbers at responding to $\mathcal{R}(\cdot,\cdot,\cdot)$ queries and use the same random numbers at responding to $\mathcal{S}(\cdot,\cdot,\cdot)$ queries which take the same inputs.

ORACLE $H_1(id)$
  if $\nexists\ L_1[id]$ **then**
    flip a coin $a \in \{0,1\}$ such that
      $\Pr[a=0]=p$ and $\Pr[a=1]=1-p$
    pick a random $\rho \in \mathbb{Z}_q^*$
    $L_1[id] \leftarrow \langle a, \rho\rangle$
  else $\langle a, \rho\rangle \leftarrow L_1[id]$
  if $a = 0$ **then return** g^ρ
  else return $(g^\beta)^\rho$
ORACLE $H_2(id, c, x)$
  if $\nexists\ L_2[\langle id, c, x\rangle]$ **then**
    pick a random $\sigma \in \mathbb{Z}_q^*$
    $L_2[\langle id, c, x\rangle] \leftarrow \sigma$
  return $L_2[\langle id, c, x\rangle]$
ORACLE $H_3(\omega)$
  if $\nexists\ L_3[\omega]$ **then**
    pick a random $\tau \in \mathbb{Z}_q^*$
    $L_3[\omega] \leftarrow \tau$
  return $L_3[\omega]$

ORACLE $\mathcal{E}(id)$
  $\langle a, \rho\rangle \leftarrow L_1[id]$
  if $a = 0$ **then return** $(g^\alpha)^\rho$ **else** abort
ORACLE $\mathcal{S}(id_A, c, id_B)$
  $\langle a_B, \rho_B\rangle \leftarrow L_1[id_B]$
  if $a_B = 1$ **then** abort
  if $\nexists L^*[\langle id_A, c, id_B\rangle]$ **then**
    pick a random $r \in \mathbb{Z}_q^*$
  else $r \leftarrow L_r[\langle id_A, c, id_B\rangle]$
  $\sigma \leftarrow H_2(id_A, c, g^r)$
  return $\langle g^r, (g^\alpha)^{\rho_B \sigma}\rangle$
ORACLE $\mathcal{R}(id_A, c, id_B)$
  $\langle a_A, \rho_A\rangle \leftarrow L_1[id_A]$; $\langle a_B, \rho_B\rangle \leftarrow L_1[id_B]$
  if $a_A = 1$ **then** abort
  pick a random $r \in \mathbb{Z}_q^*$; $\sigma \leftarrow H_2(id_A, c, g^r)$
  if $a_B = 0$ **then** $\tau \leftarrow H_3(\hat{e}((g^\alpha)^r, g^{\rho_B \sigma}))$
  else $\tau \leftarrow H_3(\hat{e}((g^\alpha)^r, (g^\beta)^{\rho_B \sigma}))$
  $L_r[\langle id_A, c, id_B\rangle] \leftarrow r$
  return $\langle g^r, (g^\alpha)^{\rho_A \tau}\rangle$

Figure 6: Simulations of the oracles $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$, H_1, H_2 and H_3.

A.1 Proof of Theorem 4.1

Proof sketch. Let $\mathcal{A}$ be a polynomial-time algorithm that attacks the privacy of the scheme 4.1. We construct a probabilistic polynomial-time algorithm $\mathcal{B}$ that given a BDH instance $\langle g, g^\alpha, g^\beta, g^\gamma \rangle$, computes $\hat{e}(g,g)^{\alpha\beta\gamma}$ by using $\mathcal{A}$ as a subroutine.

ADVERSARY $\mathcal{B}(g, g^\alpha, g^\beta, g^\gamma)$
$L_1 \leftarrow \emptyset$; $L_2 \leftarrow \emptyset$; $L_3 \leftarrow \emptyset$; set the system public key $P \leftarrow \langle g, g^\alpha \rangle$.
$\mathcal{A}$, given P, adaptively queries $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$, H_1, H_2 and H_3, and $\mathcal{A}$ outputs $(id_A^*, c_0, c_1, id_B^*)$.
recover $\langle a_B, \rho_B \rangle$ from the list $L_1[id_B^*]$.
if $a_B = 0$ **then** abort.
give $\mathcal{A}$ the challenge $\mathtt{R} \leftarrow \langle g^\gamma, z \rangle$ for a random $z \in \mathbb{G}_1$.
$\mathcal{A}$ issues additional queries, and $\mathcal{A}$ outputs a guess b'.
if $L_2 = \emptyset$ or $L_3 = \emptyset$ **then** abort.
recover σ_b from the list $L_2[\langle id_A^*, c_b, g^\gamma \rangle]$; pick a random index ω of the list L_3.
return $\omega^{(\rho_B \sigma_b)^{-1}}$ as the solution $\hat{e}(g,g)^{\alpha\beta\gamma}$.

The reason is that $\mathcal{A}$ should have issued a query $\hat{e}((g^\alpha)^\gamma, {pk_B^*}^{\sigma_b})$ to the random oracle H_3, where $pk_B^* = (g^\beta)^{\rho_B}$ and $\sigma_b = H_2(id_A^*, c_b, g^\gamma)$. Therefore, with probability 1/2 the list L_3 is defined for an index $\omega = \hat{e}((g^\alpha)^\gamma, (g^\beta)^{\rho_B \sigma_b})$. If $\mathcal{B}$ picks this index from the list L_3 then

$$\omega^{(\rho_B \sigma_b)^{-1}} = \hat{e}((g^\alpha)^\gamma, (g^\beta)^{\rho_B \sigma_b})^{(\rho_B \sigma_b)^{-1}} = \hat{e}(g,g)^{\alpha\beta\gamma} .$$

We defer the analysis of $\mathcal{B}$'s success probability to the full version of the paper. □

A.2 Proof of Theorem 4.2

Proof sketch. Let $\mathcal{A}$ be a polynomial-time algorithm that attacks the anonymity of the scheme 4.1. We construct a probabilistic polynomial-time algorithm $\mathcal{B}$ that given a BDH instance $\langle g, g^\alpha, g^\beta, g^\gamma \rangle$, computes $\hat{e}(g,g)^{\alpha\beta\gamma}$ by using $\mathcal{A}$ as a subroutine.

ADVERSARY $\mathcal{B}(g, g^\alpha, g^\beta, g^\gamma)$
$L_1 \leftarrow \emptyset$; $L_2 \leftarrow \emptyset$; $L_3 \leftarrow \emptyset$; set the system public key $P \leftarrow \langle g, g^\alpha \rangle$.
$\mathcal{A}$, given P, adaptively queries $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$, H_1, H_2 and H_3, and $\mathcal{A}$ outputs $(id_A^0, id_A^1, c^*, id_B^0, id_B^1)$.
recover $\langle a_B^0, \rho_B^0 \rangle$ from the list $L_1[id_B^0]$ and $\langle a_B^1, \rho_B^1 \rangle$ from $L_1[id_B^1]$.
if $a_B^0 = 0$ and $a_B^1 = 0$ **then** abort.
else pick a random $a_B^{b_2}$ such that $a_B^{b_2} = 1$ ($b_2 \in \{0,1\}$)
give $\mathcal{A}$ the challenge $\mathtt{R} \leftarrow \langle g^\gamma, z \rangle$ for a random $z \in \mathbb{G}_1$
$\mathcal{A}$ issues additional queries, and $\mathcal{A}$ outputs a pair of guesses (b_1', b_2').
if $L_2 = \emptyset$ or $L_3 = \emptyset$ **then** abort.
recover $\sigma_A^{b_1}$ from the list $L_2[\langle id_A^{b_1}, c^*, g^\gamma \rangle]$; pick a random index ω of the list L_3.
return $\omega^{(\rho_B^{b_2} \sigma_A^{b_1})^{-1}}$ as the solution $\hat{e}(g,g)^{\alpha\beta\gamma}$.

The reason this works is that $\mathcal{A}$ should have issued a query $\hat{e}((g^\alpha)^\gamma, {pk_B^{b_2}}^{\sigma_A^{b_1}})$ to the random oracle H_3, where $pk_B^{b_2} = (g^\beta)^{\rho_B^{b_2}}$ and $\sigma_A^{b_1} = H_2(id_A^{b_1}, c^*, g^\gamma)$. Therefore, with probability 1/4 the list L_3 is defined for an index $\omega = \hat{e}((g^\alpha)^\gamma, (g^\beta)^{\rho_B^{b_2}\sigma_A^{b_1}})$. If $\mathcal{B}$ picks this index from the list L_3 then

$$\omega^{(\rho_B\sigma_b)^{-1}} = \hat{e}((g^\alpha)^\gamma, (g^\beta)^{\rho_B^{b_2}\sigma_A^{b_1}})^{(\rho_B^{b_2}\sigma_A^{b_1})^{-1}} = \hat{e}(g,g)^{\alpha\beta\gamma} \ .$$

We defer the analysis of $\mathcal{B}$'s success probability to the full version of the paper. □

A.3 Proof of Theorem 4.3

Proof sketch. Let $\mathcal{F}$ be a polynomial-time algorithm that forges reward of the scheme 4.1. We construct a probabilistic polynomial-time algorithm $\mathcal{C}$ that given a CDH instance $\langle g, g^\alpha, g^\beta\rangle$, computes $g^{\alpha\beta}$ by using $\mathcal{F}$ as a subroutine.

ADVERSARY $\mathcal{C}(g, g^\alpha, g^\beta)$

- $L_1 \leftarrow \emptyset$; $L_2 \leftarrow \emptyset$; $L_3 \leftarrow \emptyset$; set the system public key $P \leftarrow \langle g, g^\alpha\rangle$.
- $\mathcal{F}$, given P, adaptively queries to $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, $\mathcal{R}(\cdot,\cdot,\cdot)$, H_1, H_2 and H_3, and $\mathcal{F}$ outputs $(\langle x^*, y^*\rangle, id_A^*, c^*, id_B^*)$.
- recover $\langle a_A^*, \rho_A^*\rangle$ from the list $L_1[id_A^*]$, and $\langle a_B^*, \rho_B^*\rangle$ from the list $L_1[id_B^*]$.
- **if** $a_A^* = 0$ or $a_B^* = 1$ **then** abort.
- recover σ^* from the list $L_2[\langle id_B^*, c^*, x^*\rangle]$.
- compute $\omega^* \leftarrow \hat{e}(x^*, (g^\alpha)^{\rho_B^*\sigma^*})$ and recover τ^* from the list $L_3[\omega^*]$.
- **return** $y^{*(\rho_A^*\tau^*)^{-1}}$ as the solution $g^{\alpha\beta}$.

If $\mathcal{F}$ successfully forges a reward $\langle x^*, y^*\rangle$, $\mathcal{C}$ has the equation $\hat{e}(g, y^*) = \hat{e}(g^\alpha, {pk_A^*}^{H_3(\hat{e}(x^*, {sk_B^*}^{\sigma^*}))})$, where $pk_A^* = (g^\beta)^{\rho_A^*}$ and $sk_B^* = (g^\alpha)^{\rho_B^*}$. Therefore, it knows that for $\tau^* = H_3(\hat{e}(x^*, (g^\alpha)^{\rho_B^*\sigma^*}))$,

$$\hat{e}(g, y^*) = \hat{e}(g^\alpha, (g^\beta)^{\rho_A^*\tau^*}) \ ,$$

and that $y^{*(\rho_A^*\tau^*)^{-1}}$ is the solution $g^{\alpha\beta}$ to the CDH instance $\langle g, g^\alpha, g^\beta\rangle$.

We defer the analysis of $\mathcal{C}$'s success probability to the full version of the paper. □

A.4 Proof of Theorem 4.4

Proof sketch. Let $\mathcal{F}$ be a polynomial-time algorithm that forges signature of the scheme 4.1. We construct a probabilistic polynomial-time algorithm $\mathcal{C}$ that given a CDH instance $\langle g, g^\alpha, g^\beta\rangle$, computes $g^{\alpha\beta}$ by using $\mathcal{F}$ as a subroutine.

ADVERSARY $\mathcal{C}(g, g^\alpha, g^\beta)$

- $L_1 \leftarrow \emptyset$; $L_2 \leftarrow \emptyset$; set the system public key $P \leftarrow \langle g, g^\alpha\rangle$.
- $\mathcal{F}$, given P, adaptively queries to $\mathcal{E}(\cdot)$, $\mathcal{S}(\cdot,\cdot,\cdot)$, H_1 and H_2, and $\mathcal{F}$ outputs $(\langle u^*, v^*\rangle, id^*, c^*)$.
- recover $\langle a^*, \rho^*\rangle$ from the list $L_1[id^*]$.
- **if** $a^* = 0$ **then** abort.
- recover σ^* from the list $L_2[\langle id^*, c^*, u^*\rangle]$.
- **return** $v^{*(\rho^*\sigma^*)^{-1}}$ as the solution $g^{\alpha\beta}$.

If $\mathcal{F}$ successfully forges a signature $\langle u^*, v^* \rangle$, $\mathcal{C}$ has the equation $\hat{e}(g, v^*) = \hat{e}(g^\alpha, pk_{id^*}^{\sigma^*})$, where $pk_{id^*} = (g^\beta)^{\rho^*}$ and $\sigma^* = H_2(id^*, c^*, u^*)$. Therefore, it knows that

$$\hat{e}(g, v^*) = \hat{e}(g^\alpha, (g^\beta)^{\rho^* \sigma^*}) \ ,$$

and that $v^{*(\rho^* \sigma^*)^{-1}}$ is the solution $g^{\alpha\beta}$ to the CDH instance $\langle g, g^\alpha, g^\beta \rangle$.

We defer the analysis of $\mathcal{C}$'s success probability to the full version of the paper. $\square$