ARSA: An Attack-Resilient Security Architecture for Multihop Wireless Mesh Networks

Yanchao Zhang, *Member, IEEE*, and Yuguang Fang, *Senior Member, IEEE*

***Abstract*—Multihop wireless mesh networks (WMNs) are finding ever-growing acceptance as a viable and effective solution to ubiquitous broadband Internet access. This paper addresses the security of WMNs, which is a key impediment to wide-scale deployment of WMNs, but thus far receives little attention. We first thoroughly identify the unique security requirements of WMNs for the first time in the literature. We then propose ARSA, an attack-resilient security architecture for WMNs. In contrast to a conventional cellular-like solution, ARSA eliminates the need for establishing bilateral roaming agreements and having real-time interactions between potentially numerous WMN operators. With ARSA in place, each user is no longer bound to any specific network operator, as he or she ought to do in current cellular networks. Instead, he or she acquires a universal pass from a third-party broker whereby to realize seamless roaming across WMN domains administrated by different operators. ARSA supports efficient mutual authentication and key agreement both between a user and a serving WMN domain and between users served by the same WMN domain. In addition, ARSA is designed to be resilient to a wide range of attacks. We also discuss other important issues such as incontestable billing.**

***Index Terms*—Authentication, denial-of-service (DoS), key agreement, roaming, security, wireless mesh networks (WMNs).**

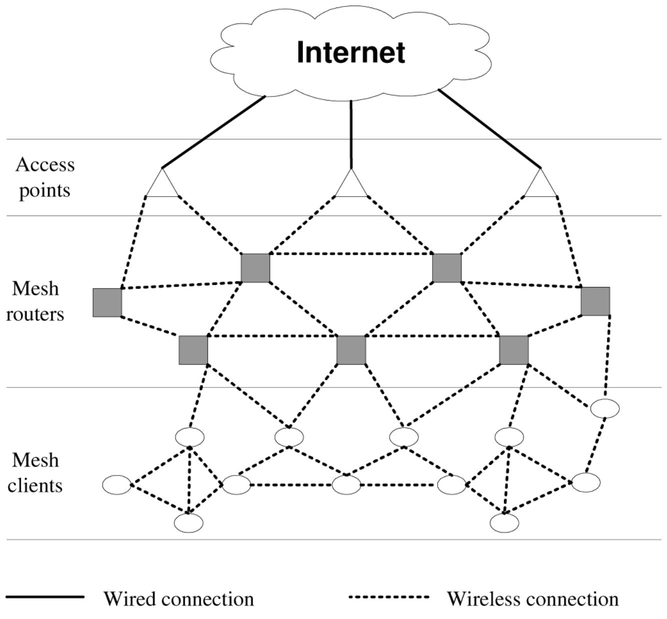


Fig. 1. A typical three-tiered WMN architecture.

I. Introduction

MULTIHOP wireless mesh networks (WMNs) are increasingly recognized as ideal solutions to ubiquitous last-mile high-speed Internet access. A typical WMN has a layered structure, as shown in Fig. 1. The first layer consists of access points (APs) which are high-speed wired Internet entry points. At the second layer, stationary mesh routers form a multihop backbone via long-range high-speed wireless techniques such as WiMAX [1]. The wireless backbone connects to wired APs at some mesh routers through high-speed wireless links. It provides multihop wireless backhaul between wired APs and mesh clients (i.e., end users) at the lowest layer.[1] Mesh clients, while at rest or in motion, can assess the network either by a direct wireless link to a nearby mesh router or by a chain of other clients to a mesh router out of reach. WMNs represent a unique marriage of the ubiquitous coverage of wide-area cellular networks with the ease and the speed of local-area Wi-Fi networks. Other notable advantages of WMNs include low deployment costs, self-configuration and self-maintenance, good scalability, high robustness, and so on [2]. Consequently, WMNs have sparkled a surge of research, development, and standardization activities, of which we refer to [2] for a comprehensive survey.

Security is one of the main barriers to wide-scale deployment of WMNs, but has gained little attention so far. The necessity for security in large-scale WMNs can be best illustrated by the following example. Suppose David wishes to retrieve some important documents from his corporate network back in Miami via a local WMN in Philadelphia, where he is on a business stay. On the one hand, the serving WMN has to corroborate the identity of David to avert fraudulent use of network resources; on the other hand, David might as well want to authenticate the serving WMN to prevent an attacker from impersonating a legitimate WMN to obtain confidential information from him. Other security concerns may include the location privacy of David, passive eavesdropping, denial-of-service (DoS) attacks, and so forth. We will dwell on the security requirements of WMNs in Section II-A.

The security of nomadic users and the serving wireless networks has been studied extensively in the past. Elegant solutions are available in the contexts of Global System for

Manuscript received September 15, 2005; revised March 30, 2006. This work was supported in part by the U.S. Office of Naval Research under Young Investigator Award N000140210464, and in part by the U.S. National Science Foundation under Grant ANI-0093241 (CAREER Award) and Grant DBI-0529012.

Y. Zhang is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: yczhang@njit.edu).

Y. Fang is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: fang@ece.ufl.edu).

Digital Object Identifier 10.1109/JSAC.2006.877223

[1]We use "client" and "user" as synonyms throughout the paper. We will not distinguish the user and the device either.

Mobile Communications (GSM) [3], Personal Communication Systems (PCSs) [4], Universal Mobile Telecommunication System (UMTS) [5], [6], and Mobile IP networks [7], among others. Despite their differences in specifics, these schemes all depend on a home/foreign-domain model. Specifically, each user has a home network domain, where he[2] is registered on a long-term basis and account information is maintained. Each time the user roams into a foreign network domain, his home domain is contacted for his credentials to authenticate him. Subsequently, the foreign domain reports the amount of service assessed by the user to his home domain which, in turn, pays the foreign domain and charges the user an amount commensurate with his usage. We argue that such solutions are less suitable for future large-scale WMNs due to at least the following reasons.

First, a bilateral service level agreement (SLA) has to be set up between each pair of network operators to permit user roaming between them. Establishing such SLAs may be a relatively easy task in cellular networks, where the operators are comparatively limited in number. Due to the easy-deployment nature of WMNs, however, the future large-scale WMNs are expected to comprise numerous WMN domains, each administrated by an independent operator [2]. Unlike a cellular operator often of a nationwide or larger scale, a WMN operator may be on a community, section, metro, or larger scale. Consequently, the number of WMN operators will be much larger than that of cellular operators. This renders it less feasible to establish pairwise bilateral SLAs among them.

Second, the above solutions all involve a potentially time-consuming and expensive execution of an authentication protocol among a user, his home domain and the foreign domain. As the user base grows large, the overall network authentication signaling overhead would be significant. In addition, in view of the high-speed wireless link, the authentication latency may be unacceptable for some short-lived data applications. Assume, for example, that a mesh client connects to a mesh router via an 802.11 a/g link with a raw rate up to 54 Mb/s. It may take the client just a couple of seconds to download several tens of MP3 music files. This makes it highly desirable to minimize the authentication delay.

Third, under conventional solutions, mesh routers will become very attractive targets and network entry points for DoS or distributed DoS (DDoS) attacks. For example, an attacker continuously sends fake authentication requests to a mesh router which, in turn, has to contact the home domains of the impersonated or even nonexistent users. If lots of collusive attackers launch this type of attack simultaneously, the resulting authentication signaling traffic will severely interfere with normal network signaling and data traffic.

Finally, conventional solutions fail to take into consideration the multihop communication paradigm featured by WMNs, as well as the communication security among mesh clients within the coverage of a same mesh router.

The limitations of conventional solutions necessitate the development of a brand-new security architecture to cope with the unique requirements of WMNs. In this paper, we answer this important open question affirmatively by proposing ARSA, an attack-resilient security architecture for large-scale WMNs. ARSA stems from an all-too-familiar scenario in real life. A user first applies for a credit card with a bank whereby to buy goods at any merchant accepting credit cards. Merchants need not establish agreements with each other, but just need to have a trust relationship with one or a few banks that accept payments from credit-card users and pay merchants. If we regard each merchant as a distinct WMN domain, the consumption of a user at different merchants can be viewed as his roaming across various WMN domains. This natural analogy motivates us to adopt the sophisticated credit-card-based business model while designing ARSA.

The players in ARSA are brokers, users, and WMN operators whose relationship is analogous to that among a bank, a credit-card user, and a merchant. Each user acquires a *universal pass* from a broker whereby to enjoy ubiquitous WMN access. Once authenticating a pass, a WMN operator can grant access to the pass holder without fear of not being paid later. As compared with conventional home/foreign-domain solutions, ARSA does not require WMN operators to establish pairwise bilateral SLAs. Rather, each WMN operator merely needs to have an agreement with one or a few brokers whose number is considered much smaller than that of global WMN operators. In addition, mutual authentication and key agreement (AKA) between a mesh client and the serving WMN domain just involve local interactions without the real-time involvement of the corresponding broker. This is particularly beneficial for reducing authentication signaling overhead and latency. Furthermore, ARSA supports efficient pairwise AKA among mesh clients present in the same WMN domain. ARSA is also designed to be resilient to various attacks, including the *location privacy* attack, the *denial-of-access* attack, the *bogus-beacon flooding* attack, and the *bandwidth-exhaustion* attack.

As far as we know, our ARSA is the first attempt to address the security of WMNs. It provides a solid foundation on which to solve other security issues in WMNs such as secure routing and medium access control (MAC). Since the research and development of WMNs are still in their very early stage, we believe that ARSA has a high potential of becoming an important component of future large-scale WMNs.

The rest of this paper is organized as follows. Section II describes the unique security requirements of WMNs and the basics of ID-based cryptography (IBC) on which we build ARSA. Next, we present the network architecture and some system models, followed by a detailed illustration of the AKA process. In Section V, we identify a few severe attacks against WMNs and provide the related countermeasures. We then discuss several other important issues in Section VI and end with conclusion and future work.

II. Preliminaries

A. Security Requirements of WMNs

Throughout this paper, we refer to the combination of the multihop wireless backbone, the wired APs, and any other WMN operator equipments, as the *infrastructure*. We also use the term "mesh" to indicate a subnet comprising a mesh router

[2]No gender implication.

and its covered mesh clients. From a high-level point of view, we identify the following security requirements of WMNs.

- *Infrastructure security*: This means the security of signaling and data traffic transmitted over the infrastructure.
- *Network access security*: This indicates the communication security between a mesh client and a mesh router. It may also involve the communication security among mesh clients served by the same mesh router, if the route between a client and a router is in multiple hops.
- *Application security*: This refers to the security of mesh clients' concrete data applications.

Among them, infrastructure security is relatively easy to achieve since the infrastructure is under the full control of a WMN operator and the network elements of the infrastructure are typically stationary. Application security can also be easily achieved via high-layer security mechanisms such as IPsec, TLS, or VPNs. By contrast, network access security is much more difficult to ensure than the other two. One major reason is that mesh routers are designed to accept open access requests by most likely unknown mesh clients. Other notable causes include open access to the wireless channels and the dynamic network topology caused by the mobility of mesh clients. For lack of space, we focus on investigating network access security in this work, and leave the exploration of the other issues as future work.

With respect to network access security, we recognize the following specific requirements, which are, however, not necessarily a complete list.

1) *Router-client authentication*: A mesh router should authenticate a requesting client to prevent unauthorized network access. The client should also authenticate the router to shun bogus mesh routers of attackers.
2) *Router-client key agreement*: The mesh router and the client should establish a shared key to encrypt and authenticate radio messages transmitted between them.
3) *Client–client authentication*: This is required when one client forwards another's traffic to and from the mesh router. In general, each client should only help other legitimate ones to get proper remuneration later.
4) *Client–client key agreement*: If needed, two mesh clients should establish a shared key whereby to encrypt and authenticate the traffic between them.
5) *Location privacy*: No entity other than a mesh client himself and a responsible location management authority (if any) should know both the real identity and the current location of the mesh client.
6) *Signaling authentication*: The signaling data broadcast by a mesh router should always be authenticated to be distinguishable from those announced by an attacker.
7) *Service availability*: A mesh router must be protected from DoS attacks and offer always available services.
8) *Incontestable billing*: A mesh client should just pay what he ought to pay, while a WMN operator, as well as those clients forwarding traffic for others, receives the amount commensurate with the offered service.
9) *Secure routing*: The routing protocol used inside a mesh should be secured against attacks.
10) *Secure MAC*: The MAC protocol employed within a mesh must be resilient to attacks.

We do not have the ambition in this paper to satisfactorily address all these requirements. Rather, we concentrate on solving the first seven issues and will outline a feasible billing scheme in Section VI. These efforts will offer a solid foundation for addressing the rest two issues.

B. Basics of IBC

As the cryptographic foundation of ARSA, IBC [8] is receiving extensive attention as a powerful alternative to traditional certificate-based cryptography (CBC). Its main idea is to make an entity's public key directly derivable from his publicly known identity information such as his e-mail address. IBC thus completely eliminates the need for public-key distribution realized via conventional public-key certificates. The recently rapid development of IBC is made possible by the application of an intriguing *pairing* technique outlined below.

Let $\mathbb{G}_1$ denote a cyclic additive group of some large prime order q and $\mathbb{G}_2$ a cyclic multiplicative group of the same order. Assume that the discrete logarithm problem (DLP) is hard[3] in both $\mathbb{G}_1$ and $\mathbb{G}_2$. For us, a pairing is a map $\mathcal{F} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ such that for all $P, Q \in \mathbb{G}_1$ and all $c, d \in \mathbb{Z}_q^*$

$$\begin{aligned}\mathcal{F}(cP, dQ) &= \mathcal{F}(cP, Q)^d = \mathcal{F}(P, dQ)^c \\ &= \mathcal{F}(P, Q)^{cd}. \end{aligned} \quad (1)$$

Note that $\mathcal{F}$ is also *symmetric*, i.e., $\mathcal{F}(P, Q) = \mathcal{F}(Q, P)$ for all $P, Q \in \mathbb{G}_1$, which follows immediately from the bilinearity of $\mathcal{F}$ and the cyclicity of $\mathbb{G}_1$. Typically, the map $\mathcal{F}$ will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field. We refer to [9] and [10] for a more comprehensive description of how the pairing parameters should be chosen in practice for both efficiency and security.

III. System Models and Notation

In this section, we present the network, trust, and pass models adopted in our ARSA, as well as the notation used.

A. Network Model

Future large-scale WMNs are expected to consist of a large number of WMN domains of different scales. Each WMN domain is operated by an independent operator and composed of a certain number of meshes, either physically adjacent or nonadjacent. For example, a WMN operator may own meshes in multiple cities or only in one city section. WMN domains may overlap with each other, and whether or not neighboring domains are connected solely depends on operator policies.

In general, a mesh router has much more powerful computation and communication capacities and abundant other resources than regular mesh clients. It is, therefore, reasonable to assume that a mesh router sends packets in one hop to all mesh clients in its coverage. By contrast, a mesh client may transmit packets in one hop or multiple hops to a mesh router within or beyond his transmission range. As noted in [11], a single-hop downlink can be highly beneficial. First, mesh clients can save their scarce energy, as there is no need to relay

[3]It is computationally infeasible to extract the integer $x \in \mathbb{Z}_q^* = \{i \mid 1 \leq i \leq q-1\}$, given $P, Q \in \mathbb{G}_1$ (respectively, $P, Q \in \mathbb{G}_2$) such that $Q = xP$ (respectively, $Q = P^x$).

downlink packets. Second, a single-hop downlink can greatly facilitate the transmissions of control signaling packets from the mesh router to all mesh clients. Last, it renders the radio resource allocation performed by the mesh router much easier to implement. Note that, however, our ARSA can be easily extended for use in symmetric WMNs with both multihop uplinks and downlinks.

It is worth pointing out that communications to and from a mesh router will be the major traffic pattern within a mesh. This is in line with the target use of WMNs, namely, relaying end users' traffic to and from the wired Internet. Such a unique traffic pattern would significantly reduce the routing complexity from mesh clients' point of view. The reason is that they only need to maintain a route to the mesh router instead of one route to each other client in the same mesh.

To make ARSA independent of the underlying network implementations, we do not specify the MAC and routing protocols in use. Interested readers are referred to [2] for a detailed survey of candidate schemes.

B. Trust Model

The trust model of our ARSA is composed of a number of *trust domains*, each managed by a broker or WMN operator. To enjoy ubiquitous WMN access, each mesh client has to first register with at least one broker which, in turn, issues an electronic universal pass to the client. If enrolling in more than one broker, a client may accordingly own multiple passes. Each WMN operator is also required to have a trust relationship with one or a few brokers. It will grant network access to mesh clients holding valid passes issued by its trustable broker(s). In fact, one may view brokers as regular banks with which both mesh clients and WMN operators have opened accounts. We assume that brokers are fully trustable by both clients and operators, but a client and an operator usually do not play full trust on each other.

The above trust model fits in well with ubiquitous Internet access via WMNs. Mesh clients see the advantage of being able to get on-demand network access by any WMN operator. The operators are relived from the heavy burden of establishing pairwise bilateral SLAs with potentially many other operators. Instead, each of them just needs to have a trust relationship with certain broker(s) whose number is considered much smaller than that of WMN operators. Furthermore, the operators have all mesh clients as potential customers, which is in contrast to the home/foreign-domain model, where a user is locked to a specific operator once signing an agreement. The brokers can make profits by deducing fees from an operator's credit or adding fees to a client's charge. They may also impose entry or subscription fees to mesh clients and operators for participation in their trust systems.

C. Notation

We denote by $\mathcal{B}_i$ and $\mathcal{O}_i$ the ith broker and WMN operator, respectively. We use $C_{i,j}$ to indicate the unique identifier of client j enrolled in $\mathcal{B}_i$. Typically, $C_{i,j}$ is of a standard format "userName@brokerName" [12]. In addition, $R_{i,j}$ refers to the unique identifer of mesh router j of $\mathcal{O}_i$, which is of the same format "routerName@operatorName." We indicate by $\mathsf{PASS}_{C_{i,j}}$ the pass of $C_{i,j}$ and by $\mathcal{K}_{C_{i,j}}$ a pass-based key (pass-key for short), both issued by $\mathcal{B}_i$ to $C_{i,j}$. Likewise, $\mathsf{PASS}_{R_{i,j}}$ and $\mathcal{K}_{R_{i,j}}$ are used to denote the router pass and the pass-key, respectively, which $R_{i,j}$ obtains from operator $\mathcal{O}_i$. Furthermore, $(\mathsf{PASS}^{\mathcal{O}_i}_{C_{i,j}}, \mathcal{K}^{\mathcal{O}_i}_{C_{i,j}})$ refers to a temporary client (pass, pass-key) pair that $\mathcal{O}_i$ issues to a served client $C_{i,j}$.

We will also use the following cryptographic primitives. $h_k(M)$ refers to the keyed message integrity code (MIC) of message M under key k, where h indicates a fast one-way hash function such as SHA-1 [13]; $\{M\}_k$ means encrypting message M under key k via a symmetric-key algorithm; $\mathcal{E}_{\mathrm{pk}}(M)$ denotes an IBC encryption operation of message M with public key pk; $\mathcal{S}_{\mathrm{sk}}(M)$ indicates message M with its IBC signature under private key sk. We refer to [14] for a number of elegant IBC encryption and signature schemes.

D. Trust-Domain Initialization

A crucial issue in ARSA is the design of passes, through which a mesh client and a serving WMN can achieve mutual AKA. It is natural to consider using digital certificates as passes. The most commonly-used X.509 certificate [15] is, however, about 1 KB in length, which might translate to a significant bandwidth overhead incurred in transmitting them. To make as short a pass as possible, we propose to utilize the emerging IBC. For this purpose, we require the administrator of each trust domain to perform the following domain-initialization operations.

1) Generate the pairing parameters $(q, \mathbb{G}_1, \mathbb{G}_2, \mathcal{F}, P, H_1)$, where P is a generator of $\mathbb{G}_1$, and H_1 is a hash function mapping given strings to nonzero elements in $\mathbb{G}_1$.
2) Pick a random $\beta \in \mathbb{Z}_q^*$ as the $\mathsf{domain-secret}$ whereby to compute a $\mathsf{domain-public-key}$ as $P_{\mathrm{pub}} = \beta P$.

We define the public *trust-domain parameters* as follows:

$$\begin{aligned}&\mathsf{domain-params}\\ &\quad := \langle \mathsf{group-params}, \mathsf{domain-public-key}\rangle\\ &\quad := \langle (q, \mathbb{G}_1, \mathbb{G}_2, \mathcal{F}, P, H_1), P_{\mathrm{pub}}\rangle.\end{aligned}$$

The domain administrator must keep β confidential, while making $\mathsf{domain-params}$ publicly known. As Diffe–Hellman group parameters used in IPsec [16], $\mathsf{group-params}$ can be standardized by such organizations as IETF. This would make it possible to use a well-known short index in place of $\mathsf{group-params}$. In contrast, β and P_{pub} should be unique to each trust domain. Also, note that it is computationally infeasible to deduce β from the (P, P_{pub}) pair because of the difficulty of solving the DLP in $\mathbb{G}_1$ (cf. Section II-B).

It is a prerequisite in an IBC cryptosystem that two communication entities use the same $\mathsf{domain-params}$. This poses the demand for an assurance on the legitimacy of $\mathsf{domain-params}$, which is satisfied in ARSA via $\mathsf{domain-params}$ certificates. In particular, we assume that there is a trusted third party (TTP) with well-known $\mathsf{domain-params}\langle \breve{q}, \breve{\mathbb{G}}_1, \breve{\mathbb{G}}_2, \breve{\mathcal{F}}, \breve{P}, \breve{H}_1, \breve{P}_{\mathrm{pub}}\rangle$ and a private domain secret $\breve{\beta} \in \mathbb{Z}_{\breve{q}}^*$. The TTP, for instance, can publish its $\mathsf{domain-params}$ through its website. Upon request of a certificate for $\mathsf{domain-params}$, the TTP computes $\breve{\beta}\breve{H}_1(\mathsf{domain-params})$ and returns it to the requesting domain administrator. We refer to such a $\langle \mathsf{domain-params}, \breve{\beta}\breve{H}_1(\mathsf{domain-params})\rangle$ pair as a

$\mathsf{domain-params}$ certificate. For ease of presentation, we indicate by $\mathsf{domain-cert}_{\mathcal{O}_i}$ and $\mathsf{domain-cert}_{\mathcal{B}_i}$ the $\mathsf{domain-params}$ certificate of operator $\mathcal{O}_i$ and broker $\mathcal{B}_i$, respectively.

To validate a $\mathsf{domain-cert}$, one just needs to check whether

$$\breve{\mathcal{F}}(\breve{P}, \breve{\beta}\breve{H}_1(\mathsf{domain-params})) = \breve{\mathcal{F}}(\breve{P_{\text{pub}}}, \breve{H}_1(\mathsf{domain-params})). \quad (2)$$

The equation should hold for an authentic $\mathsf{domain-cert}$ because

$$\breve{\mathcal{F}}(\breve{P}, \breve{\beta}\breve{H}_1(\mathsf{domain-params})) = \breve{\mathcal{F}}(\breve{\beta}\breve{P}, \breve{H}_1(\mathsf{domain-params}))$$

by the bilinearity of $\breve{\mathcal{F}}$ (cf. Section II-B) and $\breve{P_{\text{pub}}} = \breve{\beta}\breve{P}$.

Our method of certifying $\mathsf{domain-params}$ is an application of the provably secure ID-based short-signature scheme by Boneh *et al.* [17]. Another way to certify $\mathsf{domain-params}$ is to rely on conventional public-key certificates [18]. Such $\mathsf{domain-params}$ certificates can be stored at some public directory from which they can be retrieved as needed. An alternative way is to use the Domain Name System (DNS), where the $\mathsf{domain-cert}$ of each trust domain is stored and distributed as part of its DNS record [19]. Also note that, in reality, the root TTP may be replaced by a hierarchy of TTPs, similar to the traditional public-key infrastructure (PKI), in which a higher-level TTP certifies $\mathsf{domain-params}$ of each TTP at the adjacent lower level. In this scenario, a conventional certificate-chain method [20] can be used for verifying $\mathsf{domain-params}$ certificates generated by different TTPs. For clarity and ease of presentation, however, we will just discuss the single TTP case in the rest of this paper.

E. Pass Model

There are three types of passes in ARSA: router passes (R-PASSes) issued by a WMN operator to its mesh routers, client passes (C-PASSes) provided by a broker to the registered clients, and temporary client passes (T-PASSes) given by a WMN operator to mesh clients present in its domain. In this subsection, we focus on the issuance of R-PASSes and C-PASSes, and defer the discussion on T-PASSes to Section IV.

1) Issuance of R-PASSes: We take operator $\mathcal{O}_i$ as an example to explain the issuance of R-PASSes. Prior to network deployment, $\mathcal{O}_i$ issues to each controlled router $R_{i,j}$ an R-PASS $\mathsf{PASS}_{R_{i,j}} := (R_{i,j}, \mathsf{expiry-time})$, as well as a pass-key $\mathcal{K}_{R_{i,j}} = \beta^{\mathcal{O}_i} H_1^{\mathcal{O}_i}(\mathsf{PASS}_{R_{i,j}})$ which $R_{i,j}$ keeps secret. Here, $\beta^{\mathcal{O}_i}$ is operator $\mathcal{O}_i$'s $\mathsf{domain-secret}$, and $H_1^{\mathcal{O}_i}$ is the hash function specified in $\mathsf{domain-params}_{\mathcal{O}_i}$. The freshness of $\mathsf{PASS}_{R_{i,j}}$ is controlled by the $\mathsf{expiry-time}$ field. $\mathcal{O}_i$ should send to $R_{i,j}$ a new $(\mathsf{PASS}_{R_{i,j}}, \mathcal{K}_{R_{i,j}})$ pair via a secure channel before its current one expires. Depending on $\mathcal{O}_i$'s security policies, $(\mathsf{PASS}_{R_{i,j}}, \mathcal{K}_{R_{i,j}})$ may be updated hourly, daily, weekly, or even monthly. New pairs can be sent along with other domain-related control signaling traffic to minimize the communication overhead.

In essence, $(\mathsf{PASS}_{R_{i,j}}, \mathcal{K}_{R_{i,j}})$ is a standard ID-based public and private key pair in an IBC cryptosystem. Alternatively, $\mathsf{PASS}_{R_{i,j}}$ can be designed as a conventional public-key certificate and $\mathcal{K}_{R_{i,j}}$ as the corresponding private key. As compared with a typical X.509 certificate of about 1 KB, our ID-based $\mathsf{PASS}_{R_{i,j}}$ has at most a few tens of bytes in size. The main reason is that it retains the entity identifier and $\mathsf{expiry-time}$ parts of a certificate, while dumping the most space-consuming fields, namely, a public key and the digital signature of a certification authority (CA). The merits of such ID-based passes in facilitating efficient entity AKA will be seen more clearly in Section IV.

2) Issuance of C-PASSes: To enjoy ubiquitous WMN access, each client has to first register with a desired broker, similar to applying for a credit card with a bank. Consider broker $\mathcal{B}_i$ as an example. Upon a registration request from client $j, \mathcal{B}_i$ usually needs to validate the client's personal data such as his driver's licence or social security number (SSN), as well as checking his credit status. $\mathcal{B}_i$ may also ask for a security deposit as required by its registration policy. Subsequently, $\mathcal{B}_i$ assigns to the applicant an identifier $C_{i,j}$ and a C-PASS in the form of

$$\mathsf{PASS}_{C_{i,j}} := (C_{i,j}, \mathsf{expiry-time,otherTerms}).$$

Here, $\mathsf{expiry-time}$ specifies the expiry time of $\mathsf{PASS}_{C_{i,j}}$ before which $C_{i,j}$ has to renew it if desiring to stay with $\mathcal{B}_i$. Broker $\mathcal{B}_i$ may use the $\mathsf{otherTerms}$ field to name other terms and conditions $C_{i,j}$ should comply with. For instance, it may specify the per-day spending limit of $C_{i,j}$ at any WMN domain, or the list of WMN domains $C_{i,j}$ is allowed to visit, which have cooperative agreements with $\mathcal{B}_i$.

In addition to $\mathsf{PASS}_{C_{i,j}}$, the broker issues to $C_{i,j}$ a pass-key $\mathcal{K}_{C_{i,j}} = \beta^{\mathcal{B}_i} H_1^{\mathcal{B}_i}(\mathsf{PASS}_{C_{i,j}})$, where $\beta^{\mathcal{B}_i}$ is $\mathcal{B}_i$'s $\mathsf{domain-secret}$ and $H_1^{\mathcal{B}_i}$ is the hash function specified in $\mathsf{domain-params}_{\mathcal{B}_i}$. Likewise, $(\mathsf{PASS}_{C_{i,j}}, \mathcal{K}_{C_{i,j}})$ is a standard ID-based public and private key pair. As an R-PASS, $\mathsf{PASS}_{C_{i,j}}$ is much shorter than a conventional certificate realizing the same functionalities, namely, having the same $\mathsf{otherTerms}$ field.

3) Protection and Revocation of C-PASSes: Since router passes can be easily protected, here we only concentrate on protecting and revoking user passes. Client $C_{i,j}$ may store $(\mathsf{PASS}_{C_{i,j}}, \mathcal{K}_{C_{i,j}})$ in his often-used mobile device or on a USB drive to use it on multiple devices if any. $\mathsf{PASS}_{C_{i,j}}$ can be made publicly known, while $\mathcal{K}_{C_{i,j}}$ must be kept confidential to himself. There are many possible ways to protect $\mathcal{K}_{C_{i,j}}$. An all-too-familiar method is to ask $C_{i,j}$ to enter a personal identification number (PIN) for per access to $\mathcal{K}_{C_{i,j}}$.

It is possible that a careless client loses his (pass, pass-key) pair unprotected using the PIN method. This occurs, for instance, when the client loses the mobile device or the USB drive storing his secret pair. In that case, the client should report it immediately to the broker and his liability should be limited accordingly, as it is for credit-card loss. However, it should be noted that the loss of a client (pass, pass-key) pair would cause much less severe consequences or financial loss than that of a credit card. The principle reason is that C-PASSes are not designed for purchasing regular goods of possibly high values, but specifically for buying Internet access services whose rates are becoming more and more lower.

A broker can take further measures to minimize its financial risk. For example, if a client repeatedly reports a (pass, pass-key) loss, it may refuse to issue him new secret pairs. The broker may also specify a carefully designed spending-limit in a C-PASS. Moreover, the broker may use a short C-PASS validity period, say one day, and send to a client (e.g., via e-mail) a new secret pair at the early morning of each day that is only valid for that day. Furthermore, the broker can maintain a *hot list* of C-PASSes whose holders have reported losses, or which are otherwise problematic. WMN operators can periodically download the host lists from the brokers during idle hours, and refuse to serve mesh clients whose presented C-PASSes are on the host lists. Although the last measure requires certain interactions between WMN operators and brokers, it is an offline method and still considered much more lightweight than a conventional cellular-like method, where the foreign operator has to perform real-time checking with a roaming user's home operator about his account status.

IV. Authentication and Key Agreement (AKA)

In this section, we illustrate how to utilize R-PASSes and C-PASSes to realize both router-client and client–client AKA. We also distinguish *interdomain* AKA and *intradomain* AKA. The former occurs when a client migrates from one WMN domain to another, and the latter happens while a client makes his way from one mesh to another of the same WMN domain. In addition, we make the usual assumption that interdomain migrations happen less frequently than intradomain ones. So does interdomain AKA than intradomain AKA.

A. Interdomain AKA

Without loss of generality, we take client $C_{1,1}$ and mesh router $R_{1,1}$ as an example to explain the interdomain AKA protocol, which works in the following three steps:

$$\begin{aligned}
&(\mathrm{A.1})\ R_{1,1} \rightarrow * : \mathsf{PASS}_{R_{1,1}}, \mathsf{domain{-}cert}_{\mathcal{O}_1},\\
&\qquad\qquad \mathcal{S}_{\mathcal{K}_{R_{1,1}}}(t_1, \mathrm{OtherInfo})\\
&(\mathrm{A.2})\ C_{1,1} \rightarrow R_{1,1} : \mathsf{PASS}_{C_{1,1}}, \mathcal{S}_{\mathcal{K}_{C_{1,1}}}(t_2)\\
&(\mathrm{A.3})\ R_{1,1} \rightarrow C_{1,1} : \mathsf{PASS}^{\mathcal{O}_1}_{C_{1,1}}, \mathcal{E}_{\mathsf{PASS}_{C_{1,1}}}\left(\mathcal{K}^{\mathcal{O}_1}_{C_{1,1}}\right).
\end{aligned}$$

Router $R_{1,1}$ periodically broadcasts a *beacon* (A.1) via the single-hop downlink to announce its presence. The beacon should at least include $\mathsf{PASS}_{R_{1,1}}, \mathsf{domain{-}cert}_{\mathcal{O}_1}$, and a fresh timestamp t_1 signed with its pass-key $\mathcal{K}_{R_{1,1}}$ and used to defend against message replay attacks [20]. The beacon may also contain other network service information such as the current network access fee of $\mathcal{O}_1$.

The beacon can be received by all mesh clients in router $R_{1,1}$'s coverage. Assume that client $C_{1,1}$ is currently served by a WMN domain other than $\mathcal{O}_1$. Upon receipt of (A.1), he may choose to switch to $\mathcal{O}_1$ under certain conditions. For example, he may do so if $R_{1,1}$ has a much stronger signal strength than the serving router, or the access fee of $\mathcal{O}_1$ is lower than that of the serving operator. Supposing that is the case, $C_{1,1}$ performs the following operations in sequence.

1) Check whether the difference between t_1 and his local clock time is within an *acceptance window*.[4]
2) Make sure that $\mathsf{PASS}_{R_{1,1}}$ has not expired by examining its $\mathsf{expiry{-}time}$ field.
3) Validate $\mathsf{domain{-}cert}_{\mathcal{O}_1}$ according to (2).
4) Use $\mathsf{domain{-}params}_{\mathcal{O}_1}$ to verify $\mathcal{S}_{\mathcal{K}_{R_{1,1}}}(t_1, \mathrm{OtherInfo})$ with $\mathsf{PASS}_{R_{1,1}}$ as the public key.

We need to stress that $C_{1,1}$ just needs to execute step 3 once for operator $\mathcal{O}_1$. In other words, knowing the authentic $\mathsf{domain{-}params}_{\mathcal{O}_1}$ enables him to verify the signatures of any router of $\mathcal{O}_1$. If any of the checks fails, $C_{1,1}$ considers the beacon bogus and ignores it. Otherwise, he regards $R_{1,1}$ as a legitimate router of $\mathcal{O}_1$, and then forms message (A.2), including $\mathsf{PASS}_{C_{1,1}}$ and a timestamp t_2 signed under $\mathcal{K}_{C_{1,1}}$.

As for the uplink transmission of (A.2) to $R_{1,1}$, there are two cases deserving consideration. If $R_{1,1}$ is within direct reach, $C_{1,1}$ simply sends (A.2) to $R_{1,1}$ via the single-hop uplink. The more challenging case is when $R_{1,1}$ is out of $C_{1,1}$'s transmission range. A naive solution is for $C_{1,1}$ to ask clients between himself and $R_{1,1}$, which have achieved mutual authentication with and known a uplink route to $R_{1,1}$, to help relay (A.2) to $R_{1,1}$ in a hop-by-hop fashion. This measure is, however, not quite realistic since intermediate clients are generally reluctant to forward (A.2) because of the uncertainty of getting later remuneration from the as-yet unauthenticated $C_{1,1}$. It may also introduce room for a special type of DoS attack, in which an attacker continuously sends lots of faked versions of (A.2) via innocent intermediate clients to $R_{1,1}$.

Fortunately, we can deal with the second case by harnessing the transmit power control capability of many mobile devices, i.e., the ability to vary the transmit power in steps. In particular, the radio module of $C_{1,1}$ should be able to automatically boot the transmit power just enough to send (A.2) to $R_{1,1}$ in one hop. During the postauthentication stage, the transmit power can be reduced back to the normal level so that $C_{1,1}$ may send packets to $R_{1,1}$ in multiple hops. In doing so, he cannot only save his battery power, but also help increase spatial concurrency and frequency reuse, as is shown in [21].

Since brokers are relatively fewer in number, it is reasonable to assume that $R_{1,1}$ can acquire and verify the $\mathsf{domain{-}params}$ certificates of all the brokers (including $\mathcal{B}_1$) in advance. An alternative solution is to let $C_{1,1}$ append $\mathsf{domain{-}cert}_{\mathcal{B}_1}$ to (A.2). Once learning the authentic $\mathsf{domain{-}params}_{\mathcal{B}_1}$, router $R_{1,1}$ shall be able to verify the signatures by all the registered clients of $\mathcal{B}_1$. Upon receiving (A.2), $R_{1,1}$ first checks that $\mathsf{PASS}_{C_{1,1}}$ is not on the hot list of $\mathcal{B}_1$ (cf. Section III-E3). It then carries out actions analogous to what $C_{1,1}$ did. If all the inspections are successful, $R_{1,1}$ determines that $C_{1,1}$ is a legitimate registered client of broker $\mathcal{B}_1$ it trusts.

After authentication of $C_{1,1}$, router $R_{1,1}$ contacts its domain administrator to acquire the following data:

$$\begin{cases} \mathsf{PASS}^{\mathcal{O}_1}_{C_{1,1}} := (C^{\mathcal{O}_1}_{1,1}, \mathsf{expiry{-}time}) \\ \mathcal{K}^{\mathcal{O}_1}_{C_{1,1}} = \beta^{\mathcal{O}_1} H^{\mathcal{O}_1}_1(\mathsf{PASS}^{\mathcal{O}_1}_{C_{1,1}}) \end{cases}.$$

[4]This can be a fixed-size time interval, e.g., 10 ms or 20 s, preset to account for the maximum message transit and processing time, plus clock skew.

$\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}$ will be the temporary pass (T-PASS) of $C_{1,1}$ in domain $\mathcal{O}_1$, where $C_{1,1}^{\mathcal{O}_1}$ is his temporary identifier and $\mathsf{expiry-time}$ indicates the expiry time of $\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}$. Next, $R_{1,1}$ sends $\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}$ in plaintext and pass-key $\mathcal{K}_{C_{1,1}}^{\mathcal{O}_1}$ encrypted under public-key $\mathsf{PASS}_{C_{1,1}}$ to $C_{1,1}$ in message (A.3).

Upon receipt of (A.3), $C_{1,1}$ first decrypts $\mathcal{K}_{C_{1,1}}^{\mathcal{O}_1}$ using his pass-key $\mathcal{K}_{C_{1,1}}$, and then checks that the equation $\mathcal{F}^{\mathcal{O}_1}(\mathcal{K}_{C_{1,1}}^{\mathcal{O}_1}, P^{\mathcal{O}_1}) = \mathcal{F}^{\mathcal{O}_1}(H_1^{\mathcal{O}_1}(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}), P_{\text{pub}}^{\mathcal{O}_1})$ holds. Here, $\mathcal{F}^{\mathcal{O}_1}, P^{\mathcal{O}_1}$ and $P_{\text{pub}}^{\mathcal{O}_1}$ are extracted from $\mathsf{domain-params}_{\mathcal{O}_1}$. The check should succeed for a valid $(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}, \mathcal{K}_{C_{1,1}}^{\mathcal{O}_1})$ pair due to the following equations:

$$\begin{aligned}\mathcal{F}^{\mathcal{O}_1}\left(\mathcal{K}_{C_{1,1}}^{\mathcal{O}_1}, P^{\mathcal{O}_1}\right) &= \mathcal{F}^{\mathcal{O}_1}\left(\beta^{\mathcal{O}_1} H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}\right), P^{\mathcal{O}_1}\right)\\ &= \mathcal{F}^{\mathcal{O}_1}\left(H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}\right), \beta^{\mathcal{O}_1} P^{\mathcal{O}_1}\right)\\ &= \mathcal{F}^{\mathcal{O}_1}\left(H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}\right), P_{\text{pub}}^{\mathcal{O}_1}\right).\end{aligned}$$

The second line is due to the bilinearity of $\mathcal{F}^{\mathcal{O}_i}$, and the third line holds because $P_{\text{pub}}^{\mathcal{O}_1} = \beta^{\mathcal{O}_1} P^{\mathcal{O}_1}$. After a successful check, $C_{1,1}$ saves $(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}, \mathcal{K}_{C_{1,1}}^{\mathcal{O}_1})$ for subsequent use as his temporary credential in domain $\mathcal{O}_1$. Router $R_{1,1}$ and its domain administrator may record the mapping between $\mathsf{PASS}_{C_{1,1}}$ and $\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}$ if needed. We will soon show the usefulness of such temporary credentials in both intradomain client-router authentication and client–client authentication.

After a successful three-way handshake, $R_{1,1}$ and $C_{1,1}$ can establish a shared key as

$$\begin{aligned}&K_{R_{1,1},C_{1,1}}\\ &\quad= \mathcal{F}^{\mathcal{O}_1}\left(\mathcal{K}_{R_{1,1}}, H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}\right)\right)\\ &\quad= \mathcal{F}^{\mathcal{O}_1}\left(H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{R_{1,1}}\right), H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}\right)\right)^{\beta^{\mathcal{O}_1}}\\ &\quad= \mathcal{F}^{\mathcal{O}_1}\left(H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}\right), H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{R_{1,1}}\right)\right)^{\beta^{\mathcal{O}_1}}\\ &\quad= \mathcal{F}^{\mathcal{O}_1}\left(\mathcal{K}_{C_{1,1}}^{\mathcal{O}_1}, H_1^{\mathcal{O}_1}\left(\mathsf{PASS}_{R_{1,1}}\right)\right) = K_{C_{1,1},R_{1,1}}. \qquad (3)\end{aligned}$$

The above equations hold by the bilinearity and symmetry of $\mathcal{F}^{\mathcal{O}_1}$ (cf. Section II-B). Here, $R_{1,1}$ (respectively, $C_{1,1}$) derives the shared key using the first line (respectively, fourth line) pairing computation. This key agreement method is first presented in [22], which shows that the shared key will be exclusively known to the two entities establishing it. $R_{1,1}$ and $C_{1,1}$ can then use the shared key to secure subsequent traffic between them via efficient symmetric-key algorithms.

B. Intradomain AKA

Intradomain authentication occurs when client $C_{1,1}$ moves out of the coverage area of $R_{1,1}$ into that of another router of $\mathcal{O}_1$, say $R_{1,2}$. The naive reuse of the interdomain AKA protocol is less efficient because the established trust relationship between $R_{1,1}$ and $C_{1,1}$ is not exploited. Another option is to let $R_{1,1}$ hand over the shared key $K_{R_{1,1},C_{1,1}}$ to $R_{1,2}$ via a secure channel. The purpose is to allow $R_{1,2}$ and $C_{1,1}$ to authenticate each other through a classical symmetric-key challenge-response technique [20] based on $K_{R_{1,1},C_{1,1}}$. Such an approach would cause non-negligible processing burden and communication overhead on mesh routers, especially when the user base is growing large. It is also insecure to constantly use $K_{R_{1,1},C_{1,1}}$ or session keys derived from it to secure the communication between $C_{1,1}$ and multiple or even all mesh routers of $\mathcal{O}_1$.

Fortunately, possession of $(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}, \mathcal{K}_{C_{1,1}}^{\mathcal{O}_1})$ enables $C_{1,1}$ to fulfill AKA with $R_{1,2}$ by the following efficient protocol:

$$\begin{aligned}&\text{(B.1) } R_{1,2} \to * : \mathsf{PASS}_{R_{1,2}}, \mathsf{domain-cert}_{\mathcal{O}_1},\\ &\qquad\qquad \mathcal{S}_{\mathcal{K}_{R_{1,2}}}(t_1, \text{OtherInfo})\\ &\text{(B.2) } C_{1,1} \to R_{1,2} : \mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}, t_2, h_{K_{C_{1,1},R_{1,2}}}(t_1 \,\|\, t_2).\end{aligned}$$

Similar to (A.1), message (B.1) is a beacon periodically broadcast by $R_{1,2}$ to its coverage area. Upon receipt of it, client $C_{1,1}$ learns from $\mathsf{PASS}_{R_{1,2}}$ that $R_{1,2}$ is possibly another router of $\mathcal{O}_1$. He corroborates this by carrying out operations analogous to what he did in the interdomain AKA protocol. If all the inspections succeed, $C_{1,1}$ regards $R_{1,2}$ as a legitimate router of broker $\mathcal{O}_1$, and then derives a shared key $K_{C_{1,1},R_{1,2}} = \mathcal{F}^{\mathcal{O}_1}(\mathcal{K}_{C_{1,1}}^{\mathcal{O}_1}, H_1^{\mathcal{O}_1}(\mathsf{PASS}_{R_{1,2}}))$. Then, he computes a MIC $h_{K_{C_{1,1},R_{1,2}}}(t_1 \,\|\, t_2)$ and sends it together with $\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}$ and t_2 to $R_{1,2}$ in message (B.2). Here, t_2 is a fresh timestamp and $\|$ indicates concatenation. Transmission of (B.2) can be realized in a way similar to that of (A.2).

Upon receiving (B.2), $R_{1,2}$ first checks that $\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}$ has not expired and t_2 is fresh enough. If so, it then computes a shared key as $K_{R_{1,2},C_{1,1}} = \mathcal{F}^{\mathcal{O}_1}(\mathcal{K}_{R_{1,2}}, H_1^{\mathcal{O}_1}(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}))$. According to (3), only if both $C_{1,1}$ and $R_{1,2}$ are legitimate, are $K_{C_{1,1},R_{1,2}}$ and $K_{R_{1,2},C_{1,1}}$ equal to $\mathcal{F}^{\mathcal{O}_1}(H_1^{\mathcal{O}_1}(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}), H_1^{\mathcal{O}_1}(\mathsf{PASS}_{R_{1,2}}))^{\beta^{\mathcal{O}_1}}$. Router $R_{1,2}$ can make sure of this by computing a MIC $h_{K_{R_{1,2},C_{1,1}}}(t_1 \,\|\, t_2)$. If the result matches with what $C_{1,1}$ sent, it thinks of $C_{1,1}$ as a legitimate client who has been authenticated by a peer router.

The intradomain AKA protocol is more efficient than the interdomain one in both computation and communication. This is desirable because intradomain AKA needs to be done much more frequently than interdomain AKA. Note that, if $\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}$ has expired, $C_{1,1}$ has to execute the interdomain AKA protocol with $R_{1,2}$.

C. Client–Client AKA

One significant advantage of WMNs over wireless LANs lies in the multihop communication paradigm extending the network coverage. This, however, poses the demand for mutual authentication among mesh clients present in the same mesh. By client-client authentication, we mean that two mesh clients ascertain that each other is served by the same WMN domain. This is important, for example, because each client should only forward packets to the mesh router for those legitimate. Otherwise, he might get unpaid for his packet forwarding service which consumes his precious battery power. Two clients might as well wish to set up a shared key whereby to secure the data and signaling traffic between them.

The introduction of temporary client credentials greatly eases client-client AKA. The reason is that possession of an authentic temporary credential can serve as the proof that the holder has been authenticated by the current WMN domain.

Consider, for example, clients $C_{1,1}$ and $C_{2,1}$ which are registered with brokers $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. Suppose both have finished interdomain AKA with the same or different routers of operator $\mathcal{O}_1$. As a result, $C_{1,1}$ has $(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}, \mathcal{K}_{C_{1,1}}^{\mathcal{O}_1})$ and $C_{2,1}$ owns $(\mathsf{PASS}_{C_{2,1}}^{\mathcal{O}_1}, \mathcal{K}_{C_{2,1}}^{\mathcal{O}_1})$. Once actively exchanging or passively learning (e.g., from routing messages) the T-PASS of each other, they can derive the same shared key $K_{C_{1,1},C_{2,1}} = \mathcal{F}^{\mathcal{O}_1}(H_1^{\mathcal{O}_1}(\mathsf{PASS}_{C_{1,1}}^{\mathcal{O}_1}), H_1^{\mathcal{O}_1}(\mathsf{PASS}_{C_{2,1}}^{\mathcal{O}_1}))^{\beta^{\mathcal{O}_1}}$, similar to what $C_{1,1}$ and $R_{1,1}$ did in (3). Subsequently, they can fulfill mutual authentication with many classical symmetric-key challenge-response authentication techniques [20]. For instance, $C_{1,1}$ can send to $C_{1,2}$ a challenge r_1 encrypted with $K_{C_{1,1},C_{1,2}}$. If $C_{1,2}$ can report a correct response, say $(r_1+1), C_{1,1}$ declares the authentication of $C_{2,1}$ successful. In much a similar way, $C_{2,1}$ can authenticate $C_{1,1}$.

Owning an authentic temporary credential permits a client to achieve mutual AKA with all the other clients served by the same WMN domain. Also note that, unlike router-client AKA, client–client AKA can be done on demand, e.g., when two clients become neighbors, or one is helping the other deliver traffic to the mesh router. In addition, client–client AKA is expected to occur even more frequently than intradomain AKA. This is mainly due to the dynamic client join to and leave from a mesh, as well as the frequent uplink route changes caused by mobility of mesh clients or many other reasons. In light of this, our ID-based T-PASSes clearly have substantial advantages over their much longer certificate-based alternatives whose transmissions may incur a significant communication overhead.

V. Security Enhancements

Up to now, we have detailed the router-client and client–client AKA procedures based on router and client passes. The protocols presented are perfectly secure against both client and router impersonation attacks. In this section, we describe several other severe attacks against WMN access and present corresponding countermeasures. These defense mechanisms also serve as answers to security requirements five to seven introduced in Section II-A.

A. Location Privacy Attack

Anonymity and location privacy are of growing concern to end users [23]. In particular, mesh clients would usually prefer to travel *incognito*, thereby remaining anonymous to both visited WMN domains and potential eavesdroppers. In our ARSA, if a mesh client uses a fixed C-PASS while roaming, it will be possible for some attackers or vicious WMN operators to track his movements and whereabouts. We refer to such an attack as the *location privacy* attack.

Constancy and uniqueness of client identifiers are the root cause of the location privacy attack. Consider client $C_{i,j}$ as an example. As mentioned in Section III-C, $C_{i,j}$ is a standard network access identifier (NAI) [12] of format $\mathrm{ID}_{C_{i,j}}@\mathrm{ID}_{\mathcal{B}_i}$. To defend against the location privacy attack, we obviously have to ensure the confidentiality of client-name $\mathrm{ID}_{C_{i,j}}$ that is unique in domain $\mathcal{B}_i$. A straightforward solution would be to use dynamically changing aliases in place of the fixed $\mathrm{ID}_{i,j}$. One may think of also hiding the identity of broker $\mathcal{B}_i$, i.e., broker-name $\mathrm{ID}_{\mathcal{B}_i}$, as a higher-level anonymity requirement. A serving WMN domain, however, often needs to know the enrolling broker of a client. This conflict renders it unlikely to have a lightweight solution to ensuring broker anonymity. As far as we know, the only possible solution appears in [23]. In this approach, there exists a central *clearinghouse* or a mix network trusted by all brokers and WMN operators. Aliases are assigned to brokers so that a mesh client can reference his enrolling broker by an alias; it is then left up to the central clearinghouse to resolve broker aliases. Considering the infrastructure complexity related to this proposal, we currently do not feel it worthwhile to guarantee the anonymity of brokers. What we need is merely an efficient way to generate unlinkable aliases for mesh clients.

Again, we use $C_{i,j}$ as an example to explain our solution. We require that broker $\mathcal{B}_i$ have a long-enough key $\Gamma_{\mathcal{B}_i}$ which it keeps secret. The alias it generates for client $C_{i,j}$ is of an encrypted form $\mathsf{alias}_{C_{i,j}} = \{\mathrm{ID}_{C_{i,j}}, \mathrm{rand}, h_{\Gamma_{\mathcal{B}_i}}(\mathrm{ID}_{C_{i,j}} \| \mathrm{rand})\}_{\Gamma_{\mathcal{B}_i}}$, where rand denotes a random number. Consequently, $\mathsf{PASS}_{C_{i,j}}$ takes a new form, $(\mathsf{alias}_{C_{i,j}}@\mathrm{ID}_{\mathcal{B}_i}, \mathsf{expiry-time}, \mathsf{otherTerms})$. Hereafter, we refer to such a C-PASS as an alias C-PASS and the corresponding pass-key as an alias pass-key. Upon registration with $\mathcal{B}_i$, client $C_{i,j}$ is armed with multiple alias (C-PASS, pass-key) pairs, which he uses in a random fashion while roaming across WMN domains.

The use of random numbers in encryption results in unlinkable aliases. In particular, aliases for the same client are always different and an alias discloses no information about the true identity of the client. In addition, compromise of a client's alias neither compromises aliases of others nor reveals previous aliases of the same client. Therefore, the alias method provides adequate protection against the location privacy attack. It is also a stateless solution in that a broker need not book the aliases it generated. To make sure of the true identity of a client, it merely needs to perform one simple decryption of a presented alias, as well as a MIC check.

It is a must to periodically issue new alias (C-PASS, pass-key) pairs to client $C_{i,j}$. For this purpose, broker $\mathcal{B}_i$ gives a shared key $h_{\Gamma_{\mathcal{B}_i}}(\mathrm{ID}_{C_{i,j}})$ to $C_{i,j}$ during his registration. Subsequently, it uses the shared key to encrypt new alias (C-PASS, pass-key) pairs for $C_{i,j}$ who, in turn, can decrypt them for subsequent use. As for the alias update frequency, there is a tradeoff between degree of location privacy protection and alias update overhead. On the one hand, if each alias (C-PASS, pass-key) pair is used only once, we can achieve a high level of resilience to the location privacy attack. This, however, is achieved at the cost of demand for very frequent alias updates, which translate to great communication and computation overhead, *vice versa*. In practice, a good balance should be made between these two competing factors.

B. Bogus-Beacon Flooding Attack

Beacons periodically broadcast by a mesh router and processed by mesh clients place a fundamental role in ensuring the proper operation of a mesh. It is, therefore, important to guarantee the authenticity of beacons. Otherwise, an attacker may launch the *bogus-beacon flooding* attack by flooding a mesh with a lot of bogus beacons for all kinds of vicious motives. In previous intradomain and interdomain AKA protocols, a mesh router digitally sign all the beacons before sending them out to provide an assurance about their authenticity. Since beacons are usually sent in very short intervals (e.g., every 100 ms as in the IEEE 802.11b), performing continuous signature verifications will be too great a burden for common mesh clients with limited computational resources. This serves as motivation for a more lightweight yet effective solution.

We deal with this attack by a hierarchical one-way hash-chain technique, which is a modified version of the well-known Lamport's one-time-password scheme [24]. Consider router $R_{1,1}$ as an example. Assume that it broadcasts a beacon every δ ms. We also define a *super beacon interval* as a time period lasting $mn\delta$ ms, where m and n are both positive integers. With our technique in place, each beacon (A.1) from $R_{1,1}$ will take the following new form:

$$\langle \mathsf{PASS}_{R_{1,1}}, \mathsf{domain{-}cert}_{\mathcal{O}_1}, \mathrm{OtherInfo}, \mathcal{S}_{\mathcal{K}_{R_{1,1}}}(t_s \| \delta \| a_1), x, a_x, b_{x,1}, h_{a_x}(b_{x,1}), y, b_{x,y}, h_{b_{x,y}}(\text{all previous fields})\rangle$$

Here, t_s indicates the starting time of a super beacon interval; x and y are both integers such that $1 \leq x \leq m$ and $1 \leq y \leq n$; $a_x = h(a_{x+1})$ for each $x \in [1, m-1]$, where a_m is picked by $R_{1,1}$ at random; $b_{x,y} = h(b_{x,y+1})$ for each $x \in [1, m]$ and $y \in [1, n-1]$, where each $b_{x,n}$ is randomly chosen by $R_{1,1}$. Due to the one-way feature of the hash function h, if a_m is chosen randomly, given a_x it is computationally infeasible to find a_{x+1}, while given a_{x+1}, it is computationally efficient to derive a_x. Therefore, we can use the chain of values $\{a_x \mid 1 \leq x \leq m\}$ as one-time keys. The same argument applies to each chain $\{b_{x,y} \mid 1 \leq y \leq n\}$, where $b_{x,y}$ is used to compute a keyed MIC of beacon $(x-1)n + y$ of a super beacon interval. By contrast, a_x is used to calculate a keyed MIC of the initial value $b_{x,1}$ to guarantee its authenticity. To help understanding, Fig. 2 illustrates a 5-by-5 hierarchical hash chain.

Suppose client $C_{1,1}$ hears such a beacon. Let us first consider the case that $C_{1,1}$ has not fulfilled mutual authentication with router $R_{1,1}$. $C_{1,1}$ first needs to authenticate $R_{1,1}$ by performing the operations given in Section IV-A. Note that the required timestamp t_1, i.e., the beacon sending time, can be easily deduced as $t_1 = t_s + (x-1)n\delta + y\delta$. If all the checks succeed, $C_{1,1}$ then verifies that $a_1 = h^{(x-1)}(a_x)$, where $h^{(s)}(M)$ means applying the hash function h iteratively to message M for s times and $h^{(0)}(M) = M$. If so, he calculates $h_{a_x}(b_{x,1})$ compared with what is in the beacon. If the two values match, he proceeds to check the equality of $b_{x,1}$ to $h^{(y-1)}b_{x,y}$. If they are equal, $C_{1,1}$ uses $b_{x,y}$ to computed a keyed MIC of proper beacon fields and, if the result matches what he received, considers the beacon authentic. Finally, he stores the superinterval parameter triplet (t_s, δ, a_1), and sets $c_a \leftarrow x, c_b \leftarrow y, a_{c_a} \leftarrow a_x$, and $b_{c_a,c_b} \leftarrow b_{x,y}$ for later use. Other operations remain the same as those of the aforementioned interdomain or intradomain AKA protocol.

Now, we consider the case that $C_{1,1}$ and $R_{1,1}$ have authenticated each other. This means that $C_{1,1}$ has known an authentic superinterval parameter triplet of $R_{1,1}$. Upon receiving a beacon, $C_{1,1}$ first checks that the contained superinterval parameter triplet is different from what it stores, which might

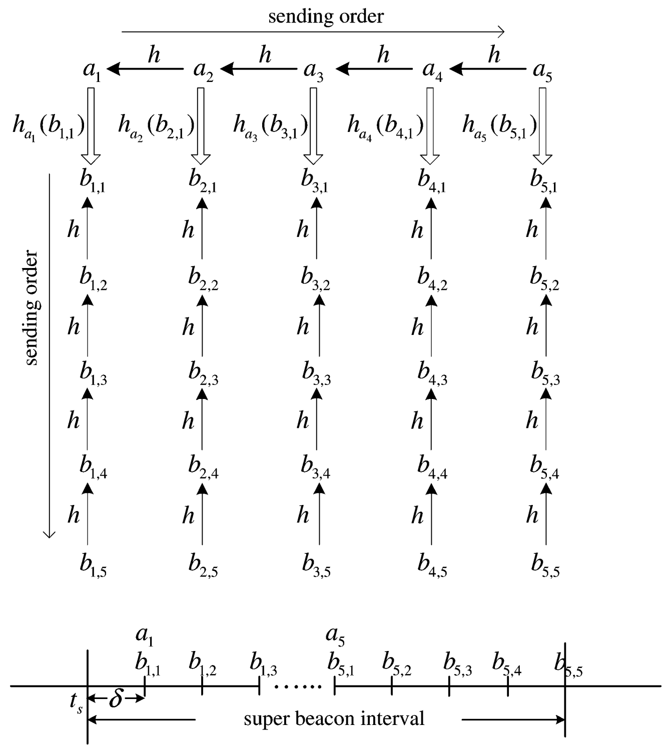


Fig. 2. An exemplary 5-by-5 hierarchical one-way hash chain.

be possible if he loses track of beacons. If so, he does the operations described above to first verify the superinterval parameters, and then authenticate the beacon. Otherwise, he first checks that $c_a \leq x$ and $c_b < y$, and then that the difference between $t_1 = t_s + (x-1)n\delta + y\delta$ and his local clock time is within an acceptance window. These checks are necessary for withstanding beacon replay attacks. If they are successful, $C_{1,1}$ further distinguishes two cases. If $a_{c_a} = a_x$, he merely checks that $b_{c_a,c_b} = h^{(y-c_b)}(b_{x,y})$ and, if so, sets $c_b \leftarrow y$ and $b_{c_a,c_b} \leftarrow b_{x,y}$. Otherwise, he needs to verify in sequence that $a_{c_a} = h^{(x-c_a)}(a_x)$, $h_{a_x}(b_{x,1})$ is equal to the MIC in the beacon, and $b_{x,1} = h^{(y-1)}(b_{x,y})$. If all the checks succeed, he computes a keyed MIC over proper beacon fields using $b_{x,y}$. Only when the result matches what is in the beacon, does he consider the beacon authentic and update $c_a \leftarrow x, c_b \leftarrow y, a_{c_a} \leftarrow a_x$, and $b_{c_a,c_b} \leftarrow b_{x,y}$.

A new super beacon interval begins either when $R_{1,1}$ has used $b_{m,n}$ or when it has updated its $(\mathsf{PASS}_{R_{1,1}}, \mathcal{K}_{R_{1,1}})$ pair.[5] In either case, it selects a random a_m and $b_{x,n}$'s for all $x \in [1, m]$, based on which to compute a new signature $\mathcal{S}_{\mathcal{K}_{R_{1,1}}}(t_s \| \delta \| a_1)$ broadcast in the next beacon.

The hash-chain technique greatly reduces the computational load of both mesh routers and clients because moderately expensive signature operations are replaced with hash operations which are usually several orders of magnitude faster. In particular, $R_{1,1}$ just needs to generate a signature at the start of each super beacon interval, rather than each time sending a beacon; each client accordingly merely performs a signature verification per super beacon interval instead of for each received beacon. The concrete performance gains are closely related to the hash-

[5]The latter case usually occurs much less frequently than the former.

chain-length parameters m, n, which, in turn, are constrained by the maximum memory the router allocates for this purpose. Generally speaking, the larger m and n, the more performance gains we can have, and *vice versa*. For instance, assume that the beacon interval is $\delta = 100$ ms, $m = 40$ and $n = 1000$, meaning a super beacon interval of about 67 min. It takes the router one signature generation and $(m-1)+m(n-1)+m+mn = 80039$ hash operations in total to generate one-time keys and keyed MICs in beacons. This is in contrast to the 40 000 IBC signature generations if the hash-chain technique is not used. In practice, a mesh router will have enough space to allow for much larger m, n values, hence meaning potentially more substantial performance gains.

In addition, the generation of $\{b_{x,y} \,|\, 1 \leq y \leq n\}$ can be deferred until the values of $\{b_{x-1,y} \,|\, 1 \leq y \leq n\}$ are almost used up. This may be desirable for lowering the storage complexity. For $\{a_x \,|\, 1 \leq x \leq m\}$ and each $\{b_{x,y} \,|\, 1 \leq y \leq n\}$, there is a computation-storage tradeoff with respect to hash-chain traversal. One may envision two extreme approaches for this problem, i.e., storing either only the hash-chain seed (a_m or $b_{x,n}$) or the entire chain. The first one has a relatively large online computational cost for generating each hash value, as the same sequence of values is repetitively computed. By contrast, the second method substantially reduces the computational complexity at the cost of high storage complexity. Researchers have recently investigated ways to optimize this computation-storage tradeoff. Interested readers are referred to [25] and [26] for a thorough treatment of this issue.

C. Denial-of-Access (DoA) Attack

A *denial-of-access* (DoA) attack is one in which an attacker sends a large number of bogus authentication responses like (A.2) or (B.2) to a mesh router. The purpose is to exhaust its resources and render it less capable of serving legitimate clients. The router is, however, assumed to at least be able to reject bogus authentication responses and send out packets.

The *client-puzzle* approach [27], [28] is a promising countermeasure against the DoA attack. The idea is quite simple. When there is no evidence of attack, a router processes authentication replies normally. Under a suspected DoA attack, the router requires that a solution to a cryptographic puzzle be attached to each authentication response. Only when the solution is correct will the router commit resources to process the response, which involves moderately expensive public-key operations. Typically, solving a client puzzle requires a brute-force search in the solution space, while solution verification is trivial. Therefore, an attacker must have access to abundant resources to be able to quickly compute a large enough number of puzzle solutions in line with his sending rate of bogus authentication responses. By contrast, although puzzles slightly increase legitimate clients' computational load when the router is under attack, they are still able to obtain network access as if there were no DoA attack. The commonly used puzzles include CPU-bound puzzles [27], [28] and memory-bound puzzles [29]. The former impose a number of computational steps to generate a solution, while the latter aim to impose similar puzzle-solving delays on clients with even different computation power. Due to space limitations, we will just demonstrate the use of CPU-bound puzzles because they are relatively easy to generate and understand. We leave the exploration of memory-bound puzzles as our future work.

With the client-puzzle approach and the aforementioned hash-chain technique, the interdomain AKA protocol given in Section IV-A is modified as follows:

$$\begin{aligned}
&(\mathrm{A.1}')\ R_{1,1} \rightarrow * : \mathsf{PASS}_{R_{1,1}}, \mathsf{domain{-}cert}_{\mathcal{O}_1}, \mathrm{OtherInfo},\\
&\qquad \mathcal{S}_{\mathcal{K}_{R_{1,1}}}(t_s \| \delta \| a_1), x, a_x, b_{x,1}, h_{a_x}(b_{x,1}), y, b_{x,y},\\
&\qquad \boldsymbol{N_{R_{1,1}}}, \boldsymbol{L_{R_{1,1}}}, h_{b_{x,y}}(\text{all previous fields})\\
&(\mathrm{A.2}')\ C_{1,1} \rightarrow R_{1,1} : \mathsf{PASS}_{C_{1,1}}, \mathcal{S}_{\mathcal{K}_{C_{1,1}}}(t_2), \boldsymbol{N_{C_{1,1}}}, \boldsymbol{X_{C_{1,1}}}\\
&(\mathrm{A.3}')\ R_{1,1} \rightarrow C_{1,1} : \mathsf{PASS}^{\mathcal{O}_1}_{C_{1,1}}, \mathcal{E}_{\mathsf{PASS}_{C_{1,1}}}\left(\mathcal{K}^{\mathcal{O}_1}_{C_{1,1}}\right).
\end{aligned}$$

The puzzle we use is similar to that of [28], consisting of $N_{R_{1,1}}$ and $L_{R_{1,1}}$ sent in beacon (A.1$'$). $N_{R_{1,1}}$ is a random nonce created and changed by $R_{1,1}$ periodically. We refer to such a period as a *puzzle interval*. $L_{R_{1,1}}$ is a 1-byte value and called the puzzle indicator. Only when there is evidence of the DoA attack does $R_{1,1}$ set the highest bit of $L_{R_{1,1}}$ to ask for puzzle solutions. In that case, the rest of the seven bits of $L_{R_{1,1}}$, denoted by $\lfloor L_{R_{1,1}} \rfloor_7$, determines the *puzzle difficulty*.

Upon receipt of the beacon, if the highest bit of $L_{R_{1,1}}$ is zero, client $C_{1,1}$ just performs the operations described before. Otherwise, he has to additionally derive a solution to the presented puzzle. He does so by first generating a random client nonce $N_{C_{1,1}}$, and then performing a brute-force search for a string $X_{C_{1,1}}$, such that the $\lfloor L_{R_{1,1}} \rfloor_7$ bits of the hash result $h(\mathsf{PASS}_{R_{1,1}} \| \mathsf{PASS}_{C_{1,1}} \| N_{R_{1,1}} \| N_{C_{1,1}} \| X_{C_{1,1}})$ are zeros. The $(N_{C_{1,1}}, X_{C_{1,1}})$ pair is a puzzle solution and returned to router $R_{1,1}$ in message (A.2$'$). If h is a good one-way hash function such as SHA-1 [13], the average number of hash operations for finding a puzzle solution is $2^{\lfloor L_{R_{1,1}} \rfloor_7}$. It is also worth noting that, since router and client passes are used in solving the puzzle, it is unlikely that the same puzzle solution can be used for other routers and clients.

After receiving (A.2$'$), router $R_{1,1}$ first checks that client $C_{1,1}$ has not previously submitted a correct puzzle solution with the same $N_{C_{1,1}}$ under the same $N_{R_{1,1}}$. Message (A.2$'$) is simply dumped if containing a replayed puzzle solution. Otherwise, $R_{1,1}$ verifies the puzzle solution by recomputing the hash to see if the $\lfloor L_{R_{1,1}} \rfloor_7$ bits of the result are all zeros. Only if the solution is correct, does it continue processing (A.2$'$) according to the previous description.

Now, we discuss the choice of puzzle parameters. To prevent an attacker from precomputing puzzle solutions, the router nonce $N_{R_{1,1}}$ must be random enough to be unpredictable. We believe that a 64-bit $N_{R_{1,1}}$ is long enough for this purpose. Also, the nonce interval should be relatively short, say one minute, to lower the risk that an attacker precomputes solutions for the same $N_{R_{1,1}}$ and $L_{R_{1,1}}$, but not be too short so as to leave a client enough time to solve the puzzle. It is possible that a legitimate client submits a solution for a puzzle interval that just ended. To allow this, there should be a short overlap between two adjacent puzzle intervals, during which the router accepts correct puzzle solutions for both intervals. Router $R_{1,1}$ can dynamically adjust the puzzle difficulty $\lfloor L_{R_{1,1}} \rfloor_7$ whose reasonable values lie between 1 and 64. The basic rule of thumb is to set $\lfloor L_{R_{1,1}} \rfloor_7$

larger when there is evidence of heavy attack and smaller otherwise. Finally, the length of a client nonce like $C_{1,1}$ can generally be shorter than that of a router nonce, but should still be long enough, say 24 bits. This is necessary to prevent an attacker from quickly exhausting all possible client nonces in the same puzzle interval with the purpose of making a router treat the puzzle solutions submitted by legitimate clients as replayed ones.

Likewise, the intradomain AKA protocol given in Section IV-B is modified as follows:

$$\begin{aligned}(\mathrm{B}.1')\ & R_{1,2} \to * : \mathsf{PASS}_{R_{1,2}}, \mathsf{domain{-}cert}_{\mathcal{O}_1}, \mathrm{OtherInfo}, \\ & \mathcal{S}_{\mathcal{K}_{R_{1,2}}}(t_s \| \delta \| a_1), x, a_x, b_{x,1}, h_{a_x}(b_{x,1}), y, b_{x,y}, \\ & \boldsymbol{N_{R_{1,2}}}, \boldsymbol{L_{R_{1,2}}}, h_{b_{x,y}}(\text{all previous fields}) \\ (\mathrm{B}.2')\ & C_{1,1} \to R_{1,2} : \mathsf{PASS}_{C_{1,1}}, t_2, \boldsymbol{N_{C_{1,1}}}, \boldsymbol{X_{C_{1,1}}} \\ & h_{K_{C_{1,1},R_{1,2}}}(t_1 \| t_2 \| \boldsymbol{N_{C_{1,1}}} \| \boldsymbol{X_{C_{1,1}}}).\end{aligned}$$

The protocol illustration is omitted here for lack of space.

D. Bandwidth-Exhaustion Attack

In a *bandwidth-exhaustion* attack, an attacker continuously sends data packets destined for a mesh router at a high data rate. Without precaution, innocent intermediate clients will waste significant resources in forwarding the attacker's packets. The attacker's traffic may also consume a significant portion of available network bandwidth, as well as interfering with legitimate clients' traffic to and from the mesh router.

We use an s-hop uplink route, starting from attacker $C_{1,1}$ through legitimate clients $C_{2,1}, \ldots, C_{s,1}$ to router $R_{1,1}$, to illustrate our countermeasures. Assume that all the clients including $C_{1,1}$ have finished mutual authentication with $R_{1,1}$ and owned an authentic temporary credential accordingly. As a result, pairwise shared keys can be established among all the clients and router $R_{1,1}$ [cf. eq. (3)]. For simplicity, we further assume that attacker $C_{1,1}$ sends out IP packets of format $\mathrm{pkt} := \langle R_{1,1}, \mathrm{data}\rangle$, where data may contain the ultimate destination to which $R_{1,1}$ should forward this packet and other upper-layer information.

An intuitive solution to the above attack is to require $C_{1,1}$ to attach to each packet s keyed MICs, computed with his pairwise keys shared with intermediate clients and $R_{1,1}$. More specifically, each packet sent by $C_{1,1}$ takes a new form,[6] $\mathrm{pkt}' := \langle \mathrm{pkt}, h_{K_{C_{1,1},C_{2,1}}}(\mathrm{pkt}), \ldots, h_{K_{C_{1,1},C_{s,1}}}(\mathrm{pkt}), h_{K_{C_{1,1},R_{1,1}}}(\mathrm{pkt})\rangle$. Upon receipt of such a packet, each intermediate client $C_{i,1}$ for $i \in [2, s]$ can verify the MIC $h_{K_{C_{1,1},C_{i,1}}}(\mathrm{pkt})$ before forwarding it to the next hop. Finally, router $R_{1,1}$ verifies $h_{K_{C_{1,1},R_{1,1}}}(\mathrm{pkt})$ before processing the packet. This method can withstand the bandwidth-exhaustion attack by an attacker not authenticated by the serving WMN domain, as his packets will not carry correct keyed MICs. In addition, if an authenticated attacker like $C_{1,1}$ follows the process correctly, router $R_{1,1}$ can slow down his traffic by economic means. Particularly, $R_{1,1}$ regards $C_{1,1}$ as a normal client with a high bandwidth demand and charges him a large amount commensurate with his traffic rate. However, the economic means fails if $C_{1,1}$ always inserts into each packet incorrect keyed MICs only for the last few hops. In doing so, his packets will always be dropped by intermediate clients before reaching $R_{1,1}$, thus $R_{1,1}$ has no way of charging $C_{1,1}$. However, $C_{1,1}$ can still effectively achieve the vicious goal of consuming network and legitimate clients' resources.

A complementary way to mitigate the bandwidth-exhaustion attack is through the aforementioned client-puzzle approach. It utilizes the fact that each served client of $R_{1,1}$ can hear the puzzle $(N_{R_{1,1}}, L_{R_{1,1}})$, and is thus able to validate puzzle solutions. In this approach, $C_{1,1}$ needs to provide $R_{1,1}$ with a puzzle solution $(N_{C_{1,1}}, X_{C_{1,1},R_{1,1}})$ satisfying the aforementioned constraint. He also has to offer a solution $(N_{C_{1,1}}, X_{C_{1,1},C_{i,1}})$ for each intermediate client $C_{i,1}$, which should satisfy that the $\lfloor L_{R_{1,1}} \rfloor_7$ bits of $h(\mathsf{PASS}_{C_{i,1}} \| \mathsf{PASS}_{C_{1,1}} \| N_{R_{1,1}} \| N_{C_{1,1}} \| X_{C_{1,1},C_{i,1}})$ are all zeros. Each such solution can be individually validated by the intended client.

If suspecting the presence of the bandwidth-exhaustion attack, router $R_{1,1}$ sets the highest bit of $L_{R_{1,1}}$ to instruct all clients within coverage to perform validations of puzzle solutions. If this occurs, each packet source like $C_{1,1}$ needs to send puzzle solutions along with data packets at a rate in line with his traffic rate. We use the well-known token-bucket approach to realize this objective. In particular, each intermediate client $C_{i,1}$ maintains a token bucket for $C_{1,1}$, essentially an integer counter of sufficient length, say 4 bytes. He adds α tokens to the bucket each time $C_{1,1}$ provides a correct puzzle solution. Each token corresponds to a traffic unit, say 1 KB, and only when there are enough tokens in the bucket, will $C_{i,1}$ forward $C_{1,1}$'s packets to the next hop after doing a MIC check. The rate-control parameter α can be dynamically adjusted to cope with the current network traffic load. Specifically, it should be set smaller when the traffic load is heavy and larger otherwise. $R_{1,1}$ can either centrally decide α conveyed to mesh clients in beacons, or let each client determine α by himself.

VI. Discussion

In this section, we discuss other issues relevant to ARSA.

A. Incontestable Billing

Billing of mobile users is conventionally realized in a way that the foreign domain reports the amount of service accessed by the user to his home domain which, in turn, bills the user and pays the foreign domain. This approach does not ensure billing incontestability, as the foreign domain may overstate the amount of service the user actually used, while the user may deny the service he requested. Under our ARSA, it is promising to achieve incontestable billing of mesh clients via an intriguing lightweight micropayment approach [30]. Consider router $R_{1,1}$ and client $C_{1,1}$ as an example. To pay $R_{1,1}$ in return for network access, $C_{1,1}$ creates a chain of hash tokens, $x_0, x_1, \ldots, x_m$, where x_m is chosen at random and $x_i = h(x_{i+1})$ for all $i \in [0, m-1]$. Each hash token is associated with a monetary value τ. Initially, $C_{1,1}$ sends a signed payment commitment $\mathcal{S}_{\mathcal{K}_{C_{1,1}}}(\tau, x_0)$ to $R_{1,1}$ which, in turn, verifies and saves the commitment. At each subsequent predetermined interval, $C_{1,1}$ releases a hash token in sequence. $R_{1,1}$ can authenticate the token by checking that it can hash to x_0 after a few hash operations. When the session terminates,

[6]There are ways to shorten the packet, which are ignored for brevity.

$R_{1,1}$ keeps the triplet $\langle \mathcal{S}_{\mathcal{K}_{C_{1,1}}}(\tau, x_0), m^*, x_{m^*} \rangle$ as a billing record, where m^* indicates the highest token index from $C_{1,1}$. Later, operator $\mathcal{O}_1$ sends the billing record to $C_{1,1}$'s enrolling broker $\mathcal{B}_1$ which, after verifying $\mathcal{S}_{\mathcal{K}_{C_{1,1}}}(\tau, x_0)$ and token x_{m^*}, pays $\mathcal{O}_1 m^* \tau$ monetary units and charges $C_{1,1}$ the same amount. In this approach, operator $\mathcal{O}_1$ cannot overcharge client $C_{1,1}$ because it is unable to fake correct tokens hashing to $x_0; C_{1,1}$ cannot deny the bill since he has signed the payment commitment and no one else can provide correct hash tokens. A thorough investigation on this issue can be found in [18].

B. Stimulating Packet Forwarding

Generally speaking, mutually strange mesh clients are unwilling to forward others' traffic to and from the mesh router to save their own resources. We thus must research how to motivate cooperation in packet forwarding to enable the highly beneficial multihop communication paradigm. This issue has been studied extensively in mobile ad hoc networks (MANETs), see, for example [31]. The existing solutions are not directly applicable to WMNs and our ARSA. We believe that it is rather feasible to reward packet forwarding also by the micropayment approach. The idea is quite simple. Each client pays the router what the router and all intermediate clients should get. The router, in turn, rewards mesh clients for forwarding all others' traffic by generating a unique chain of hash tokens for each of them. This approach is lightweight from mesh clients' viewpoint, as they just need to have a payment relationship with the router instead of each of others. Please refer to [18] for a full exploration of this issue.

C. Incremental Deployment

One of the main barriers to wide deployment and use of WMNs is the lack of a sound business model. Our ARSA affirmatively answers this problem and is highly advantageous for WMN operators, mesh clients, and brokers. As the development of the credit card system, we expect ARSA to be deployed incrementally along with WMNs. Initially, there might be only one broker, which might be an enterprising regular bank or emerging electronic money transmitter like PayPal, a few WMN operators, and a limited number of mesh clients. As time goes on, the shown benefits of ARSA would attract more and more operators to built WMNs and users to use WMN services, and increasing brokers (though still limited in number) to act as trust intermediaries.

VII. Conclusion

For the first time in the literature, this paper identifies and satisfies a number of unique security requirements of the emerging multihop WMNs. We present an attack-resilient security architecture, called ARSA, for large-scale WMNs. In contrast to a conventional cellular-like solution, ARSA is more practical and lightweight because it does not require a WMN operator to establish pairwise bilateral SLAs and interact in real-time with potentially numerous other WMN operators. ARSA is also a *homeless* solution in which each user, instead of being bound to any specific WMN operator, can get ubiquitous network access by a universal pass issued by a third-party broker. ARSA provides efficient mutual AKA not only between a user and a serving WMN domain but also between users served by the same WMN domain. In addition, it is designed to be resistant to various attacks against WMN access.

References

[1] The WiMAX Forum. [Online]. Available: http://www.wimax-forum.org

[2] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Comput. Netw.*, vol. 47, no. 4, pp. 445–487, Mar. 2005.

[3] European Telecommunications Standards Institute (ETSI), "GSM 2.09: Security Aspects." Jun. 1993.

[4] H. Lin and L. Harn, "Authentication protocols for personal communication systems," in *Proc. ACM SIGCOMM*, Cambridge, MA, Aug./Sep. 1995, pp. 256–261.

[5] 3GPP, TS 21.102, 3rd Generation Partnership Project (3GPP); Technical Specification Group (TSG) SA; 3G Security; Security Architecture, version 4.2.0, Release 4, 2001.

[6] Y. Lin and Y. Chen, "Reducing authentication signalling traffic in third-generation mobile network," *IEEE Trans. Wireless Commun.*, vol. 2, no. 3, pp. 493–501, May 2003.

[7] C. Perkins, "IP mobility support for iPv4," IETF, RFC 3344, Aug. 2002.

[8] A. Shamir, "Identity based cryptosystems and signature schemes," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 1984, vol. 196, Proc. CRYPTO, pp. 47–53.

[9] D. Boneh and M. Franklin, "Identify-based encryption from the Weil pairing," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2001, vol. 2139, Proc. CRYPTO, pp. 213–229.

[10] P. Barreto, H. Kim, B. Bynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2002, vol. 2442, Proc. CRYPTO, pp. 354–368.

[11] M. Jakobsson, J.-P. Hubaux, and L. Buttyan, "A micro-payment scheme encouraging collaboration in multi-hop cellular networks," in *Proc. 7th Int. Conf. Financial Cryptography*, Gosier, Guadeloupe, Jan. 2003, pp. 15–33.

[12] B. Aboda and M. Beadles, "The network access identifier," IETF, RFC 2486, Jan. 1999.

[13] *Digital Hash Standard*, Federal Information Processing Standards Publication 180-1, NIST, Apr. 1995.

[14] R. Dutta, R. Barua, and P. Sarkar, "Pairing-based cryptography: A survey," Cryptology ePrint Archive Rep. 2004/064, 2004.

[15] *Authentication Framework*, ITU-T Recommendations X.509, ITU, Geneva, 1989.

[16] D. Harkins and D. Carrel, "The Internet key exchange (IKE)," IETF, RFC 2409, Nov. 2003.

[17] D. Boneh, B. Lynn, and H. Shacham, "Short signature from the Weil pairing," in *Proc. ASIACRYPT*, Gold Coast, Australia, Dec. 2001, pp. 514–532.

[18] Y. Zhang and Y. Fang, "A secure authentication and billing architecture for wireless mesh networks," *Wireless Netw.* [Online]. Available: http://winet.ece.ufl.edu/~zhang/mesh-WINET.pdf, to be published

[19] D. Smetters and G. Durfee, "Domain-based administration of identity-based cryptosystems for secure email and ipsec," in *Proc. 12th USENIX Security Symp.*, Washington, DC, Aug. 2003, pp. 215–229.

[20] A. Menezes, P. van Oorschot, and S. Vanston*, Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1996.

[21] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[22] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *Proc. Symp. Cryptography Inf. Security*, Okinawa, Japan, Jan. 2000, pp. 26–28.

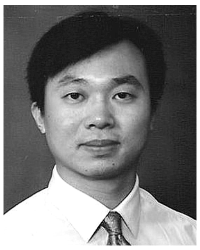
[23] G. Ateniese, A. Herzberg, H. Krawczyk, and G. Tsudik, "Untraceable mobility or how to travel incognito," *Comput. Netw.*, vol. 31, no. 8, pp. 871–884, Apr. 1999.

[24] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, pp. 770–772, Nov. 1981.

[25] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal," in *Proc. Financial Cryptography*, Southampton, Bermuda, Mar. 2002, pp. 102–119.

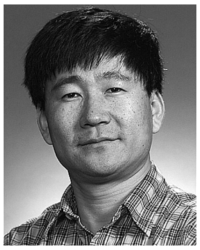
[26] Y. Sella, "On the computation-storage trade-offs of hash chain traversal," in *Proc. Financial Cryptography*, Guadeloupe, French West Indies, Jan. 2003, pp. 270–285.

[27] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *Proc. 6th Annu. Netw. Distrib. Syst. Security Symp.*, San Diego, CA, Feb. 1999, pp. 151–165.
[28] T. Aura, P. Nikander, and J. Leiwo, "DoS-resistant authentication with client puzzles," in *Proc. 8th Int. Workshop Security Protocols*, Cambridge, U.K., Apr. 2000, pp. 178–181.
[29] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, "Moderately hard, memory-bound functions," in *Proc. 10th Annu. Netw. Distrib. Syst. Security Symp.*, San Diego, CA, Feb. 2003, pp. 25–39.
[30] R. Rivest and A. Shamir, "Payword and micromint: Two simple micropayment schemes," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, vol. 1189, Proc. Int. Workshop Security Protocols, pp. 69–87.
[31] Y. Zhang, W. Lou, and Y. Fang, "A secure incentive protocol for mobile ad hoc networks," *Wireless Netw.* [Online]. Available: http://winet.ece.ufl.edu/~zhang/sip-WINET.pdf, to be published



Yanchao Zhang (S'03–M'06) received the B.E. degree in computer communications from Nanjing University of Posts and Telecommunications, Nanjing, China, in July 1999, the M.E. degree in computer applications from Beijing University of Posts and Telecommunications, Beijing, China, in April 2002, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, in August 2006.

Since September 2006, he has been an Assistant Profession in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark. His research interest include wireless and Internet security, wireless networking, and mobile computing.



Yuguang Fang (S'92–M'93–SM'99) received the B.S. and M.S. degrees in mathematics from Qufu Normal University, Qufu, Shandong, China, in 1984 and 1987, respectively, the Ph.D. degree in systems and control engineering from the Department of Systems, Control and Industrial Engineering, Case Western Reserve University, Cleveland, OH, in January 1994, and the Ph.D. degree in electrical engineering from the Department of Electrical and Computer Engineering, Boston University, Boston, MA, in May 1997.

From 1987 to 1988, he held research and teaching positions in both the Department of Mathematics and the Institute of Automation, Qufu Normal University. From September 1989 to December 1993, he was a Teaching/Research Assistant in the Department of Systems, Control and Industrial Engineering, where he held a Research Associate position from January 1994 to May 1994. He held a Postdoctoral position in the Department of Electrical and Computer Engineering, Boston University from June 1994 to August 1995. From September 1995 to May 1997, he was a Research Assistant in the Department of Electrical and Computer Engineering, Boston University. From June 1997 to July 1998, he was a Visiting Assistant Professor in the Department of Electrical Engineering, University of Texas at Dallas. From July 1998 to May 2000, he was an Assistant Professor in the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark. In May 2000, he joined the Department of Electrical and Computer Engineering, University of Florida, Gainesville, where he received early promotion to Associate Professor with tenure in August 2003, and to Full Professor in August 2005. He has published over 150 papers in refereed professional journals and conferences. His research interests span many areas including wireless networks, mobile computing, mobile communications, wireless security, automatic control, and neural networks.

Dr. Fang received the National Science Foundation Faculty Early Career Award in 2001 and the Office of Naval Research Young Investigator Award in 2002. He also received the 2001 CAST Academic Award. He is listed in *Marquis Who's Who in Science and Engineering*, *Who's Who in America*, and *Who's Who in World*. He has been actively engaged in many professional activities. He is a member of the Association for Computing Machinery (ACM). He is an Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, an Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING, an Editor for *ACM Wireless Networks*, and an Editor for the IEEE WIRELESS COMMUNICATIONS. He was an Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (Wireless Communications Series), an Area Editor for the *ACM Mobile Computing and Communications Review*, an Editor for the *Wiley International Journal on Wireless Communications and Mobile Computing*, and Feature Editor for Scanning the Literature in the *IEEE Personal Communications*. He has also been actively involved with many professional conferences such as ACM MobiCom'02 (Committee Co-Chair for Student Travel Award), MobiCom'0l, IEEE INFOCOM'06, INFOCOM'05 (Vice-Chair for Technical Program Committee), INFOCOM'04, INFOCOM'03, INFOCOM'00, INFOCOM'98, IEEE WCNC'04, WCNC'02, WCNC'00 (Technical Program Vice-Chair), WCNC'99, IEEE Globecom'04 (Symposium Co-Chair), Globecom'02, and the International Conference on Computer Communications and Networking (IC3N) (Technical Program Vice-Chair).