Software-Defined Networking Based Capacity Sharing in Hybrid Networks

Mateus A. S. Santos
and Bruno T. de Oliveira
and Cintia B. Margi
University of Sao Paulo

Bruno A. A. Nunes
and Thierry Turletti
INRIA Sophia Antipolis

Katia Obraczka
University of California at Santa Cruz

***Abstract*—This paper proposes a novel approach to capacity sharing in hybrid networked environments, i.e., environments that consist of infrastructure-based as well as infrastructure-less networks. The proposed framework is based on Software-Defined Networking (SDN) and provides flexible, efficient, and secure capacity sharing solutions in a variety of hybrid network scenarios. In this paper, we describe the challenges raised by capacity sharing in hybrid networks, describe our framework in detail and how it addresses these challenges, and discuss implementation issues. To the best of our knowledge, this is the first SDN-based capacity sharing solution that targets hybrid networks and that incorporates security as an integral part of the proposed approach.**

I. Introduction

As mobile devices equipped with multiple network interfaces become commonplace, users will expect "anywhere, anytime" connectivity regardless of location or type of network access. As a result, a major challenge facing future networks is to provide ubiquitous connectivity in a resource-efficient fashion. Efficient utilization of network resources is critical since expanding network resources at the same rate as network traffic increases is not economically viable. The need to utilize network resources efficiently is exacerbated in wireless (access) networks, where resources are inherently more constrained and traffic is expected to double every year in the next few years[1]. This trend is generally referred to as "capacity sharing" [1].

In this paper, we explore solutions to capacity sharing in wireless access networks. More specifically, we examine scenarios such as the one depicted in Figure 1 in which a wireless infrastructure-less network can be used to extend the scope of the existing infrastructure-based network. For example, a user, say "Alice", who may be temporarily without access to her network service provider, may connect to the Internet through the infrastructure-less network (in this example, through "Bob"). Participants in the infrastructure-less network may receive incentives from their service provider to serve as "gateways" to other users. Clearly, security is a major concern as existing standards (e.g., 802.1x[2]) do not provide adequate security for these types of scenarios and applications. For instance, in the particular scenario of Figure 1, "Bob" may need to authenticate "Alice" to make sure she is a legitimate user, etc. Furthermore, "Bob" should not be liable for misbehaving users (here "Alice") connecting through him. At the same time, data confidentiality should be provided to users connecting through "Bob" in order to preserve privacy.

In order to provide simple and efficient security services, we employ ID-Based Cryptography (IBC) [2], [3]. By eliminating the need for generating and managing users' certificates, IBC significantly reduces the complexity of a cryptographic system. Moreover, IBC does away with manual configuration of shared keys among backend players, which is required in protocols such as RADIUS/EAP[3]. This is achieved through key agreement procedures, which also allow users to compute shared keys. In addition, we prevent restrictions imposed by credentials based on username/password, such as delay of typing this information and the use of devices with no keyboard.

Our approach to capacity sharing in heterogeneous networks as exemplified by the scenario in Figure 1 is based on Software-Defined Networking (SDN) techniques. SDN presents an opportunity to address the challenges discussed above by programmatically controlling networks. This is achieved through the decoupling of the control– from the data–plane by: (1) removing control decisions from the forwarding hardware, (2) allowing the forwarding hardware to become "programmable" via an open interface, and (3) having a separate entity called "controller" to define by software the behavior of the network formed by the forwarding infrastructure, thereby creating a "software-defined network". OpenFlow [4] is a notable example of a SDN architecture and has been considered as the de-facto standard protocol used for communication between the controller and programmable packet forwarding devices.

Motivated by the vision of a fully-connected world in which wireless access networks extend the scope of the wired infrastructure [5], we propose a SDN-based framework for flexible, efficient, and secure capacity sharing in current and emerging hybrid network environments. To the best of our knowledge, this is the first SDN-based capacity sharing solution targeting hybrid networks. Additionally, it incorporates security as an integral part of the proposed approach. In this paper, we describe our framework in detail and showcase its use in the context of the application shown in Figure 1.

[1] http://icnp13.informatik.uni-goettingen.de/workshops/csws-13.html
[2] http://www.ieee802.org/1/pages/802.1x-2004.html.
[3] https://tools.ietf.org/html/rfc5247.

II. Network Model

We assume the network model illustrated in Figure 1. In this paper, we focus on scenarios where a user "Alice" wishes to connect to the Internet and access, for example, the World Wide Web. However, she is unable to connect to the existing network infrastructure (e.g., because she is out of range of the closest AP). Another user "Bob" advertises his gateway services providing "Alice" the option to connect to the Internet through him. Note that Alice can connect to Bob directly or through a wireless, multi-hop ad-hoc network (MANET).

A. Traditional– versus SDN-Enabled Scenario

Traditional Scenario: Assuming the ad hoc network learns to route to Bob as a gateway, and Bob allows his device to be used as a NAT box by other users, the mobile data service provider is not aware of the existence of Alice. Bob's connection is not assigned additional bandwidth, possibly harming performance; the Internet Service Provider is not able to differentiate Alice from Bob and cannot apply any QoS rules, access restriction, or any sort of policies on Alice's traffic without also impacting Bob's; furthermore, Bob will be held responsible for Alice's traffic by the service provider for any possible data overages or illegal activity.

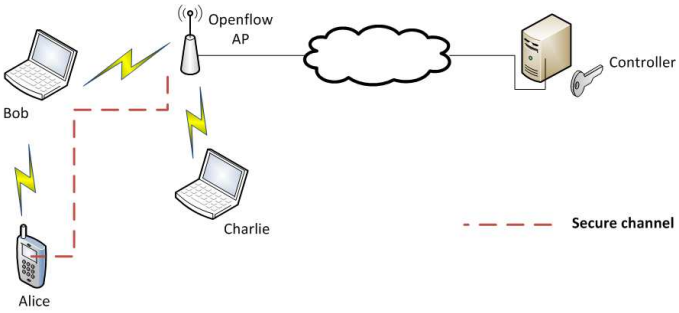

Fig. 1. Application scenario using SDN.

SDN-Enabled Scenario: When Alice joins the network, the service provider (through its SDN controller) is made aware of Alice's presence. It may then decide to offer service to Alice via Bob and provision Bob's connection accordingly. The service provider may decide to sell Alice a temporary connection plan on the spot, or Alice may have an existing contract on another device; available resources, past user behavior, or any number of factors can be used in deciding whether to offer service to Alice, and if so, what kind of service to offer. The service provider is thus able to maintain control of its network resources and how they are utilized and shared, while being granted opportunity for additional business. Alice is able to seamlessly connect to the Internet using an existing service plan or on a "pay-per-use" basis. For his part, Bob may be offered incentives by the service provider, while avoiding performance loss or being held liable for Alice's traffic.

B. Requirements

Below, we enumerate issues that should be addressed in order to enable the scenario in Figure 1. For each case, we propose a SDN-enabled solution. It is worth noting that we do not focus on a specific capacity sharing algorithm. We aim at developing a secure framework that enables a number of new applications and services, including QoS polices and capacity sharing schemes that do not rely solely on the end hosts.

1) End Device Deployment: The concept of SDN is centered around controlling forwarding policies and has been thus far put into practice in infrastructure-based networks. However, in infrastructure-less environments such as mobile ad hoc networks (MANETs) or vehicular networks (VANETs), the devices involved in data forwarding are also end devices themselves. Consequently, end devices should be able to perform the functions of SDN switches, i.e., communicate with controllers and understand how to handle forwarding rules. Since end devices in infrastructure-less networks are typically portable and do not have access to continuous power sources, the deployment need to be lightweight in terms of its code, storage, communication, and power consumption footprint.

In this work we propose the instantiation of a software module at the gateway device (in our case, Bob's device). We call it the *Switching Module (SM)*. From an implementation point of view, the SM can be instantiated as an OpenFlow software switch running on the gateway (GW) device and is responsible for forwarding incoming traffic, maintaining flow tables, and communicating with the controller when needed.

2) Gateway Device Incentive: Similar to other capacity sharing solutions, our framework assumes that end users will be incentivized to contribute to sharing network resources, which in our case will be in the form of relaying traffic for other users. Incentives, their policies and implementation are outside the scope of this paper.

3) Access Control: To control the access of network resources it is required not only to authenticate a new node X, but also to ascertain membership eligibility and bootstrap security services such as data confidentiality and authenticity. In a typical wireless scenario, access control can be achieved by the 802.1x standard, including the RADIUS/EAP protocol. In 802.1x, the authenticator is the end of the link requiring authentication. It usually operates as a pass-through, forwarding packets between the backend server and the user. The EAP framework requires that the backend server (here the controller) can only distribute cryptographic keys to the authenticator (here the GW). This is referred to as "the principle of mode independence"[4] and shows that RADIUS has lack of compatibility with non-collocated authenticator function and encryption/decryption function. Note that, in our scenario, even though the GW is able to perform authentication, it cannot decrypt packets from Alice; thus, different keys still need to be established between Alice and the other players, namely the AP and GW. In order to enable user authentication via the GW device, we propose to have the GW run an *Authentication Module (AM)* integrated with the SM. Membership eligibility and distribution of cryptographic material are performed by means of an application on top of the controller.

Considering that each participating node already possesses its unique identification ID_X and private key S_X (as defined in the Setup procedure in Section IV-B), we now describe the scenario in which a new node (e.g. Alice) will connect to an ad hoc network and access the infrastructure network via a GW device (i.e. via Bob).

[4]https://tools.ietf.org/html/rfc5247

The AM installed on Bob's device will communicate with Alice so that mutual authentication is performed. In order to do that, a node only needs the other node's identity for computing a shared key, which is then used in a *challenge/response* procedure (as defined in Section IV-C). After node admission, the computed shared key is also used for confidentiality and data authentication. The AM will then notify the controller that might accept or reject Alice's request. Note that the controller is only notified by Bob after Alice is authenticated, which protects the central entity against possible denial of service attacks.

Upon acceptance, the controller will insert the appropriate new flow table entries to Bob's SM, in order to grant Alice access to the outside network via OpenFlow protocol. Concurrently, the controller proactively sets the appropriate flow table entries in every forwarding device under its control, in Alice's data path to the Internet, allowing her access to the infrastructure of the network. This proactive setting of forwarding rules prevents all forwarding devices in the data path from the user to the Internet send *packet-in* messages to the controller, saving resources and increasing scalability.

4) Confidentiality and Data Authenticity: Even though Bob provides Alice with the service of message forwarding, he should not be able to decrypt her traffic. Thus, Alice needs to establish a secure channel with other entity. However, she might not have another peer that is capable of providing a VPN service. Our proposed architecture suppresses the need for additional VPN services and confidentiality is intrinsically included. Since all the wireless traffic should be received by the AP, it turns out that the best candidate for implementing a secure channel with Alice is the AP. That is possible by agreeing on a shared key with Alice.

It is worth noting that Bob and Alice can exchange authenticated messages without using encryption. This is due to the fact that Alice already sends encrypted messages to the AP, using Bob as the intermediate. However, some use cases require that they exchange messages confidentially. For example, direct communication between Alice and Bob either started by an application or a *handshaking* procedure.

5) Access Restriction and QoS Policies: Current OpenFlow version 1.3, already allows QoS polices to be enforced by means of creating virtual ports on the switches and applying weighted fair queuing (WFQ). In our case, when Alice joins the network, after authenticated, the controller might choose to restrict her access to certain applications (e.g. deny/restrict Bittorrent connections or more bandwidth demanding applications such as video streaming) in order to preserve, not only Bob's access, but also the access and quality of experience of other users connected via Bob in the same manner as Alice.

6) Changing Gateways: In the case of a scenario with multiple gateways, as depicted by Figure 1, another user (here "Charlie") joins the network and is able to act as a gateway. Charlie also has the SM and AM running on his device. The SM communicates the controller that Charlie has a new interface on, in ad hoc mode.

The controller now knows that Charlie might be another candidate for Alice. For example, Bob's mobility can cause low signal strength and Bob can also be disconnected if out of the AP's range or by system outage. The controller is able to set a flow table entry at Charlie's OpenFlow software switch, in order to make Charlie the responsible for forwarding Alice's traffic. Consequently, it might delete the corresponding flow table entry from Bob's software switch. Note that the corresponding entry would expire in case Bob cannot be reached by the controller.

In the aforementioned scenario, one can also consider SDN applications that exploit many wireless networks, as demonstrated by prior works [6], [7]. For example, Alice can be provided with services by two gateways simultaneouly (here, Bob and Charlie). As one can see, this is another application made possible by a SDN based architecture, not possible or very challenging and prone to errors on the traditional scenario.

III. Background

Before presenting the proposed architecture, we review the main concepts related to security services.

A. Symmetric Ciphers and MACs

Confidentiality can be provided by means of symmetric ciphers (e.g., AES [8]), which are generally more efficient than public key encryption. Data authentication can also be implemented efficiently by means of symmetric cryptography using Message Authentication Codes (MACs) (e.g., CMAC [9]). A MAC operates over a symmetric cipher for generating authentication tags, which are sent together with messages. The recipient can use its own key to calculate the expected tag; the message is accepted (i.e., authenticated) only if the calculated value matches the received tag.

The main issue of symmetric cryptography is how to share secret keys without a secure channel. Fortunately, it is possible to use public key cryptography for efficient pairwise key agreement, which then allows the use of symmetric cryptography. To that purpose, we consider the approach of ID-Based Cryptography, which we review next.

B. ID-Based Cryptography

Identity Based Cryptography (IBC) [2], [3] allows a user to calculate a public key from an arbitrary string. Using the users' identity as a public key has advantages such as: (1) there is no need to verify the public key using an online Certification Authority (C.A.); and (2) a user only needs the recipients' identities in order to calculate public keys (i.e., there is no need to ask for public keys). In addition, cryptographic protocols are simple and efficient under the paradigm of IBC.

Given that a user's public key is tied to an unique identity, the issue becomes how to obtain the corresponding secret key. A Trusted Third Party (TTP) is responsible for secret generation, which is performed by using its own secret key, also known as *master secret key*, and the public key of the target user.

Note that all secret keys can be computed by the TTP. Fortunately, in the scenario explored here, there is a synergy present between controllers and TTPs. Controllers can be considered trusted entities, since they provide interfaces to applications that perform management tasks. Thus, in the context of IBC, a controller could be responsible for generating (and possibly distributing) private keys to users in its domain.

α	OpenFlow controller
β	OpenFlow Access Point (AP)
ID_X	identity of node X
s	master secret key (controller's secret key)
S_X	private key of node X
P_X	public key of node X (derived from ID_X)
$K_{X,Y}$	key established between nodes X and Y
ctr	counter
$\texttt{authenc}(msg, k)$	authenticated encryption of msg using key k
$\texttt{enc}(msg, k)$	encryption of msg using key k
$\texttt{dec}(msg, k)$	decryption of msg using key k
mac	authentication tag
$e(\cdot, \cdot)$	pairing function

TABLE I. NOTATION.

Even though IBC was introduced by Shamir [2], it was only realized with bilinear mappings, or pairings [3], [10]. Pairings also provide practical implementation for authenticated key agreement over IBC, which is an elegant alternative to non-authenticated schemes such as the Diffie-Hellman interactive key exchange.

Pairings for Authenticated Key-Agreement

Authenticated Key-Agreement (AKA) over IBC can be implemented by means of pairings, which we informally define as follows. Let $\mathbb{G}$ be a cyclic additive group and $\mathbb{G}_T$ be a cyclic multiplicative group of the same order q, which is a positive integer. Then, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map that satisfies (1) bilinear: $e(aP, bQ) = e(P, Q)^{ab}\ \forall P, Q \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_q^*$; (2) non-degenerate: $e(P, P) \neq 1$; and (3) computable: there exists an efficient algorithm to compute $e(P, Q)\ \forall P, Q \in \mathbb{G}$. The group $\mathbb{G}$ can be implemented using a group of points on an elliptic curve and the group $\mathbb{G}_T$ using a subgroup of a finite field.

It is worth noting that the AKA procedure considered here has the main goal of avoiding public key encryption. It means that, once a key is agreed between two nodes using public key cryptography (i.e., IBC), they can use the shared key for confidentiality and data authentication (i.e., symmetric cryptography), which is very efficient.

IV. SECURE CAPACITY SHARING FRAMEWORK

In this section, we present the proposed secure capacity sharing framework. The notation used to describe our framework is presented in Table I.

A. Overview

A setup procedure should be performed before secure communication among nodes. In the setup phase, each node receives from the TTP a private key and public parameters. The latter are mainly the group order, group generators and functions such as the pairing and the mapping from identity to public key. For a detailed description, we refer to the work of Boneh and Franklin [3]. The main steps for secure network access extension using SDN, illustrated in Figure 2, are as follows:

- **Gateway discovery:** The potential gateway nodes (e.g. Bob) will send `Hello` messages periodically, announcing their gateway capabilities. The potential users (e.g. Alice), on the recipient of such `Hello` messages will choose the best GW, by sending a request message to the most suitable candidate based on metrics such as link quality or received signal strength. Other metrics can be considered, such as capacity of links connecting the GW node to the infrastructure, traffic load at the GW and etc. This information can be added to the hello messages so that a user Alice, could make a more educated decision.
- **Handshaking**: Since Bob is a GW capable node (i.e. it has the SM and AM installed and is able to perform GW duties) it responds to Alice's request and initiates a handshaking procedure for Alice's authentication;
- **AKA and user's check-in**: once Alice is authenticated, she can agree on shared keys with the AP and the controller for using symmetric cryptography. In addition, Bob sends an OpenFlow *packet-in* message, so that the controller can proactively add the new flow-table entries to the AP, Bob and the forwarding devices (i.e. OpenFlow switches) on Alice's data path towards the Internet;
- **End-to-end security**: messages can be exchanged securely using symmetric cryptography.

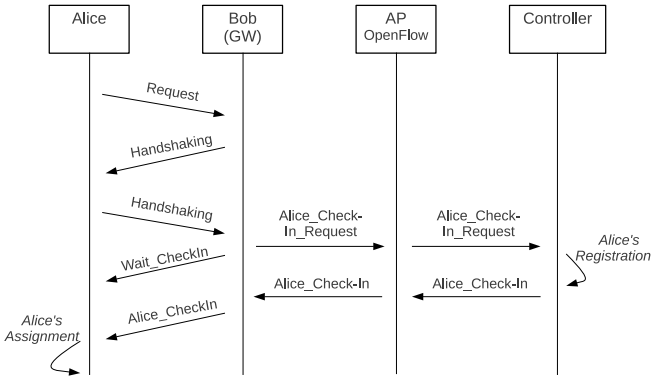

Fig. 2. High level description of the service.

B. Setup

As public keys are derived from identities, the TTP (i.e., the controller) maps the node identity, ID_X, to a point in the elliptic curve, P_X. This mapping is a public parameter, since a node is allowed to generate any node's public key. The TTP generates a master secret key s and calculates each node's private key as $S_X = sP_X$. This value should be either sent privately by the TTP or pre-deployed on the node.

C. Handshaking

In the handshaking procedure, Alice is requested to respond to a challenge, so that Bob is able to verify Alice's identity. It is worth noting that Bob and Alice should know each other's identity for exchanging authentication messages. This allows them to compute a shared key, which is used for authenticated encryption of the challenge. Thus, Bob can encrypt a message

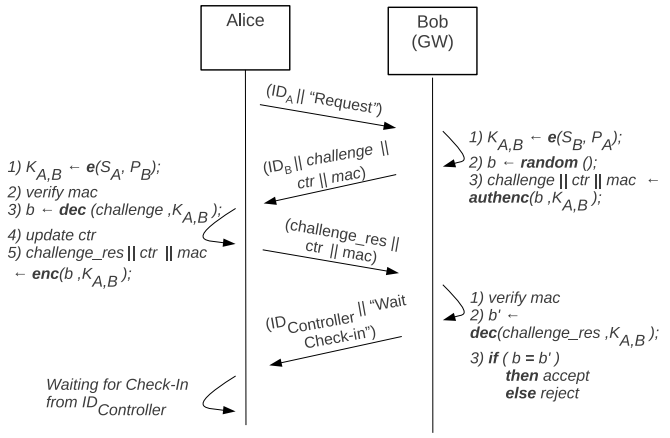

Fig. 3. Detailed handshaking procedure.

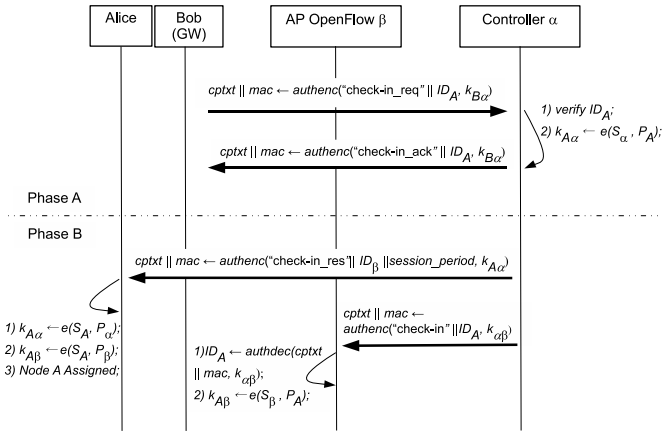

Fig. 4. Authenticated key-agreement.

to Alice, which decrypts the ciphertext to obtain the challenge that can be encrypted again and sent back to Bob. Figure 3 shows in detail this process, in which Alice and Bob use a counter in order to protect messages from *replay attacks*.

D. Authenticated Key-Agreement and User's Check-in

After the handshaking procedure, Alice needs to establish pairwise keys with the AP and the controller. It allows her to send encrypted messages that the GW cannot decrypt. In addition, Alice needs a shared key with the controller for proving her identity, which prevents a malicious Bob from pretending to be a gateway for users that actually do not exist. Such a key can also be used for `check_out` messages, which are requests for being disconnected from the GW (and not being charged for services).

We employ the SOK protocol [10] in the AKA procedures. Figure 4 illustrates the exchanged messages for key agreement between Alice and the other participants, namely the AP and Controller. Note that the purpose of such messages is not only to establish pairwise keys[5]. The message `check-in_req` has the goal of requesting the controller authorization for secure communication. If a user is not authorized by the controller, neither flows or keys will be created. That can be possible by means of an application running on top of the OpenFlow controller.

Since the node attached to the AP operates as an OpenFlow switch, it can send *packet-in* messages to the controller. Then, upon verifying the user and the associated flow, *packet-out* messages can be sent to the entities that comprise the overall application. This is illustrated in Figure 5. Security for *packet-in* and *packet-out* messages is provided by the OpenFlow standard, which is based on TLS. Thus, this procedure does not depend on the security architecture that is proposed here.

E. End-to-End Security

End-to-end security is implemented using symmetric encryption. Generally, encryption is provided with authentication, since encryption-only schemes might succumb to attacks (e.g., encryption-only configurations of IPsec [11]). For this reason, we consider authenticated encryption of messages exchanged between Alice and Bob, as well as Alice and the AP. This is illustrated in Figure 6, in which security is transparent between the GW and the AP (e.g., using WPA2).

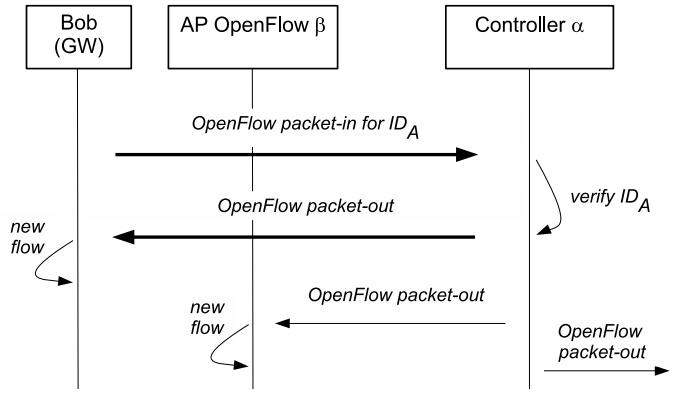

Fig. 5. OpenFlow messages for the new flow.

Moreover, regarding Alice's traffic, messages relayed to and from the AP cannot be interpreted by the GW (or by an eavesdropper). On the other hand, the GW (i.e. Bob) is able to authenticate messages it relays to the AP, which protects it from relaying unauthorized messages. To summarize, not only can the gateway authenticate messages from Alice, but also transmit authenticated encryption messages to her. The latter is used in the handshaking procedure.

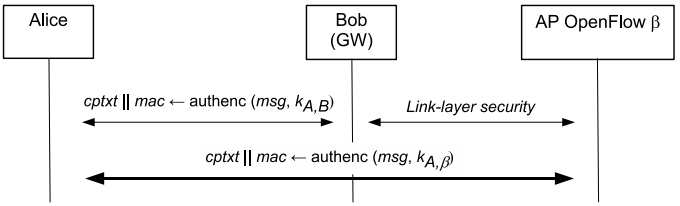

Fig. 6. End-to-end security.

V. Implementation Issues

We are currently implementing our framework atop the OpenFlow protocol leveraging our prior work on secure SMS

[5]Key agreement could be achieved in a non-interactive fashion if identities and public parameters are known.

transmission [12]. The most expensive cryptographic operation used in the scenario proposed here is the computation of the pairing function, which is used in the handshaking and key agreement procedures.

Based on our prior work, we can compute the pairing in about 1 second using a cell phone of 434 MHz and 128 MB SDRAM memory. Since this computation is only performed in the first steps of the protocol, we argue that the proposed security solution is feasible in real scenarios. Moreover, our implementation is based on Java and can easily be easily integrated into OpenFlow controllers such as Floodlight[6].

VI. Related Work

He et al. [13] employ IBC and pairings for secure handover authentication. In addition, some works propose cooperative relaying using medium access control protocols (e.g., [14]) or physical layer protocols, such as the work of Krikidis et al. [15], which also provides protection against eavesdropping.

Our proposal is based on OpenFlow [4] and, to the best of our knowledge, is the first to address security services such as admission control, data authentication and confidentiality for end users and OpenFlow software switches (the GW and the AP are software switches extended to support the security functionalities).

While previous works have examined the use of SDN in wireless environments, their scope has primarily focused on wireless infrastructure deployments (e.g., Wi-Fi, WiMAX access points). A notable example is the OpenRoads project [7] which envisioned a world in which users could freely move between wireless infrastructures while also providing support to the network provider. Other works such as [16], [17], [18] have examined OpenFlow in wireless mesh environments.

VII. Conclusions

In this paper we proposed an efficient, flexible, yet secure SDN-based framework for capacity sharing in hybrid networked environments. Our proposed framework extends the scope of the existing network infrastructure and enables a number of new applications and services for mobile users. From the network service provider's point of view, by performing efficient sharing of network resources, our framework aims at minimizing the need for over-provisioning the network.

Our framework explores the synergy between SDN controllers and TTPs, so that security initialization (e.g., key distribution) relies on existing network infrastructure. It is worth noting that such an approach requires additional mechanisms for controller coordination. In addition, given that the SDN paradigm is typically centralized, inter-domain roaming should rely on different key management schemes or protocol extensions.

As part of our ongoing research, we intend to demonstrate the feasibility of our approach by implementing it in an real testbed. We also plan to demonstrate the ability to perform seamless and secure handover as well as extend access to devices participating in multi-hop wireless networks, e.g., MANETS or VANETs.

[6]http://www.projectfloodlight.org/floodlight/

References

[1] M. Mendonca, B. A. A. Nunes, K. Obraczka, and T. Turletti, "Software defined networking for heterogeneous networks," *IEEE COMSOC MMTC E-Letter*, 2013.

[2] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO*, ser. LNCS, G. Blakley and D. Chaum, Eds. Springer Berlin Heidelberg, 1985, vol. 196, pp. 47–53.

[3] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO 2001*, ser. LNCS, J. Kilian, Ed. Springer Berlin Heidelberg, 2001, vol. 2139, pp. 213–229.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[5] B. Rais, M. Mendonca, T. Turletti, and K. Obraczka, "Towards truly heterogeneous internets: Bridging infrastructure-based and infrastructureless networks," in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*. IEEE, 2011, pp. 1–10.

[6] K.-K. Yap, T.-Y. Huang, M. Kobayashi, Y. Yiakoumis, G. Appenzeller, and N. McKeown, "Openflow demo - using all wireless networks around me." [Online]. Available: http://www.openflow.org/videos

[7] K. Yap, M. Kobayashi, R. Sherwood, T. Huang, M. Chan, N. Handigol, and N. McKeown, "Openroads: Empowering research in mobile networks," *ACM SIGCOMM Comput. Communic. Rev.*, vol. 40, no. 1, pp. 125–126, 2010.

[8] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer, 2002.

[9] NIST, *Special Publication 800-38B Recommendation for Block Cipher Modes of Operation*, National Institute of Standards and Technology, May 2005.

[10] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *Symposium on Cryptography and Information Security (SCIS00)*, 2000, p. 2628.

[11] J. Degabriele and K. Paterson, "Attacking the ipsec standards in encryption-only configurations," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, 2007, pp. 335–349.

[12] G. C. Pereira, M. A. Santos, B. T. de Oliveira, M. A. S. Jr., P. S. Barreto, C. B. Margi, and W. V. Ruggiero, "Smscrypto: A lightweight cryptographic framework for secure SMS transmission," *Journal of Systems and Software*, vol. 86, no. 3, pp. 698 – 706, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121212003056

[13] D. He, C. Chen, S. Chan, and J. Bu, "Secure and efficient handover authentication based on bilinear pairing functions," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 1, pp. 48–53, 2012.

[14] H. Adam, W. Elmenreich, C. Bettstetter, and S. Senouci, "Core-mac: A mac-protocol for cooperative relaying in wireless networks," in *GLOBECOM 2009. IEEE*, 2009, pp. 1–6.

[15] I. Krikidis, J. Thompson, and S. McLaughlin, "Relay selection for secure cooperative networks with jamming," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 10, pp. 5003–5011, 2009.

[16] A. Coyle and H. Nguyen, "A frequency control algorithm for a mobile adhoc network," in *Military Comm. and Information Systems Conference (MilCIS)*, Canberra, Australia, November 2010.

[17] P. Dely, A. Kassler, and N. Bayer, "Openflow for wireless mesh networks," in *ICCCN*. IEEE, 2011, pp. 1–6.

[18] X. Nguyen, "Software defined networking in wireless mesh network," INRIA, UNSA, MSc. Thesis, August 2012.