An Authentication Scheme for IEEE 802.11s Mesh Networks Relying on Sakai-Kasahara ID-Based Cryptographic Algorithms

Aymen Boudguiga, Maryline Laurent
Institut TELECOM, TELECOM SudParis, CNRS Samovar UMR 5157
9 rue Charles Fourier, 91011 Evry, France
Email: {Aymen.Boudguiga, Maryline.Laurent}@it-sudparis.eu

***Abstract*—Nowadays authentication in Wireless Mesh Networks (WMN) refers to the 802.1X authentication methods or a Preshared key authentication, and makes use of certificates or shared secrets. In wireless environments, the management of certificates is a cumbersome task as certificates require deploying a Public Key Infrastructure (PKI) and Certification Authorities (CA). They also require defining a certificate management policy to control the generation, transmission and revocation of certificates. During the last decade, ID-Based Cryptography (IBC) appeared as a good alternative to PKI. IBC proposes to derive the public key from the node's identity directly thanks to the use of a Private Key Generator (PKG).**

In this article, we present an authentication method relying on an ID-Based signature and encryption schemes that use the Sakai-Kasahara key construction. The resulted authentication scheme is suitable to IEEE 802.11s mesh networks and resistant to the key escrow attack.

I. Introduction

Nowadays authentication in Wireless Mesh Networks (WMN) refers to the 802.1X authentication methods or a Preshared key authentication, and makes use of certificates or shared secrets. For certificate management, there is a need to deploy a PKI for defining CAs responsible for the certificate creation, revocation and transmission. Revocation can be done through Certificate Revocation Lists (CRL) periodically issued by the CA and updated with newly revoked certificates. CRLs have to remain available on demand to the network stations (STAs) for checking validity of the certificates before use. The management of certificates and CRL is a cumbersome task, especially for wireless networks, as it is bandwidth and memory consuming.

During the last decade, researcher interest in ID-Based Cryptography (IBC) increased because they represent a good alternative to PKI. IBC was introduced by A. Shamir [1] to provide entities with public/private key pairs with no need for certificates, Certification Authorities (CA) and PKI. Shamir assumes that each entity uses its identifier as its public key. In addition, he assigned the private key generation function to a special entity which is called Private Key Generator (PKG). That is, before accessing the network, every entity has to contact the PKG to get back a smart card containing its private key. This private key is computed so it is bound to the public key of the entity. Then, IBC usage evolved thanks to the use of the Elliptic Curve Cryptography (ECC) [2]. As a consequence, new ID-Based signature schemes emerged and they differ from Shamir's method in that PKG does not rely on smart cards to store the private key and the ciphering information. Note that ID-Based cryptography requires lightweight implementations on clients. Compared to PKI certificate management, there is no need for storing certificates, and the key revocation operation is much simpler. Key revocation in ID-Based cryptography is bound to a validity period which is defined by the PKG. Interested readers might refer to the article [3] for a good comparison between PKI and ID-Based cryptography.

The main problem of ID-Based cryptography lies on PKG which fully generates the private key of every entity, and as such, is given the knowledge to perform a key escrow attack. With the usual strong assumption that PKG is a trustworthy entity like in IEEE 802.11s mesh network standards [4], this attack has never been considered seriously. Nonetheless the PKG can easily impersonate as the legitimate station (STA) or decrypt its ciphered traffic, and as such, it is worth proposing key escrow resistant solutions.

In this article, we propose to use ID-Based Signature (IBS) and Encryption (IBE) schemes with the Sakai-Kasahara method for ID-Based key construction [5], as it permits to get very fast IBS and IBE processing. In addition, we propose to use the Mesh Key Distributors (MKDs) defined in [4] as PKGs. The main idea of our authentication scheme is to make every station authenticate to an Authentication Server (AS) which delegates the station key generation to MKDs. That is, AS is only used for STA authentication while MKDs control the STA key derivation. Consequently, the risks of a Denial of Service (DoS) attack against AS are reduced because AS memory consumption during key generation is removed. We also propose to make MKD generate a partial private key for each authenticating station, so it is able to compute its own private key. In the following, we show how this partial private key generation avoids the key escrow threat on the PKG (i.e. MKD).

The article is organised as follows. After introducing the ID-Based cryptography and the IEEE 802.11s mesh architecture, in section III, we present our authentication scheme and an informal security analysis of it where the resistance to key escrow attack is explained. In addition, we explain the motivation that made us choose an IBS and IBE relying on Sakai-Kasahara key construction for our authentication scheme.

II. ID-Based Cryptography

When a station needs a private key, it provides PKG with the identity ID intended to be used for its private key computation. The PKG then derives the node's private key using some parameters which must be defined with respect to the Bilinear Diffie-Hellman problem [6]. For generating these parameters, PKG runs a Probabilistic Polynomial Time (PPT) algorithm which takes as input a security parameter k and outputs the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ and the pairing function $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_2$ in $\mathbb{G}_T$. $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups of prime order q and $\mathbb{G}_T$ is a multiplicative group of the same order q. Note that the order q is defined with respect to k such that $q > 2^k$. Generally, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of the group of points of an Elliptic Curve (EC) over a finite field and $\mathbb{G}_T$ is a subgroup of a multiplicative group of a related finite field.

The pairing function $\hat{e}$ has to be bilinear, non degenerate and efficiently computable. The non degeneracy property means that for all points $P \in \mathbb{G}_1$, $\hat{e}(P, 1_{\mathbb{G}_2}) = 1_{\mathbb{G}_T}$. In addition, for all points $Q \in \mathbb{G}_2$, $\hat{e}(1_{\mathbb{G}_1}, Q) = 1_{\mathbb{G}_T}$. If we consider a generator P of $\mathbb{G}_1$ and a generator Q of $\mathbb{G}_2$, the value $\hat{e}(P, Q) = g$ is equal to the generator of $\mathbb{G}_T$. The generator point P is used to compute another point P_{pub}. Practically this kind of bilinear mapping is derived from the Weil or Tate pairing (or any efficient pairing) [7]. In the following sections, we consider only symmetric pairings i.e. $\mathbb{G}_1 = \mathbb{G}_2$.

In addition to the definition of groups, some hash functions need to be defined in accordance to the IBE or IBS schemes that are going to be used. For example, a hash function H that verifies $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ is defined in order to transform the node's identity into an EC point. Generally, the public key of a station is computed as a hash of one of its identities and it is either a point of an elliptic curve or a positive integer. The list containing the groups $\mathbb{G}_1$ and $\mathbb{G}_2$, the bilinear mapping $\hat{e}$, the points P and P_{pub} and the hash functions form the *public elements*. These *public elements* are distributed by PKG to the network users because they are needed during the public key derivation and the cryptographic operations.

The key derivation operation starts when PKG receives the ID of the node that is requesting a private key (Figure 1). First, PKG computes the user's public key as $Pub_{ID} = H(ID)$. Then, PKG generates the corresponding private key using a local secret value $s \in \mathbb{Z}_q^*$. Note that the private key is computed as: $Priv_{ID} = f(s, Pub_{ID})$. In the most common cases, $Priv_{ID} = s \cdot Pub_{ID}$ where $Pub_{ID} \in \mathbb{G}_1$. The secret value s is also used for P_{pub} derivation from P: $P_{pub} = s \cdot P$. As such, the *public elements* are $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, g, P, P_{pub}, H_1, \ldots, H_k\}$. However, Sakai and Kasahara defined a new construction method that permits to define very efficient IBS and IBE in terms of computation time. The Sakai-Kasahara key construction scheme proposes to compute the private key as $Priv_{ID} = \frac{1}{Pub_{ID}+s}P$ where s is PKG's secret. In the following, we present Barreto et al. signature [8] and Chen et al. encryption [5] which uses this key construction scheme.

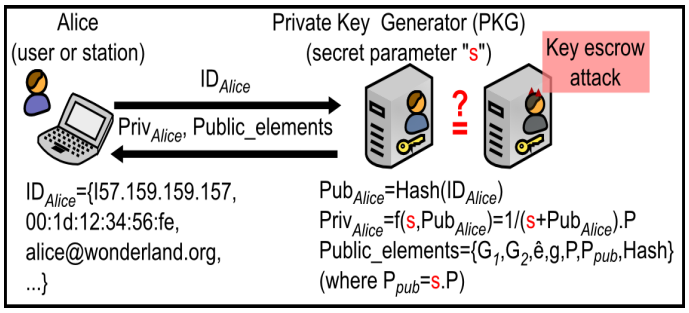

Fig. 1. ID-Based key generation.

It is clear from the aforementioned key derivation schemes that PKG knows every private key it generates itself, and as such it is able to impersonate as a private key owner by illegally generating signature or deciphering encrypted traffic. Generally to mitigate the key escrow attack, a strong assumption is made necessary that PKG is a trustworthy entity. However, in this article we propose a new method that is resistant to the key escrow attack with no need for the previous assumption. Our idea is to make PKG generate a partial private key for STA which is then able to generate its own private key. As such PKG is not able to recover the STA private key. Our solution does not only remove the key escrow attack, but also it does not introduce many changes in the used signature or encryption scheme. For more details about our authentication scheme, please refer to section III.

A. Barreto et al. Signature Scheme

Barreto et al. presented their ID-Based signature scheme (BLMQ) in 2005 [8]. BLMQ basically uses an asymmetric pairing function. However, we present it considering a symmetric pairing function. BLMQ signature scheme defines two hash functions H_1 and H_2 such that: $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_2 : \{0,1\}^* \times \mathbb{G}_T \rightarrow \mathbb{Z}_q^*$. So *BLMQ public elements* are $\{G_1, \mathbb{G}_T, q, \hat{e}, g, P, P_{pub}, H_1, H_2\}$. A user public key is computed as $Pub_{ID} = H_1(ID)$ and its corresponding private key is generated by PKG as $Priv_{ID} = \frac{1}{Pub_{ID}+s}P$. In order to sign a message M, the signer chooses a random number $k \in \mathbb{Z}_q^*$ and executes the following steps:

1) $n = g^k$
2) $h = H_2(M, n)$
3) $S = (k + h)Priv_{ID}$

The signature is formed by the pair $(S, h) \in G_1 \times \mathbb{Z}_q^*$. Then, the signature verifier has only to check the equality between h and $H_2(M, \hat{e}(S, H_1(ID)P + P_{pub})g^{-h})$.

B. Chen et al. Encryption Scheme

Chen et al. presented their ID-Based encryption scheme in [5]. They define two hash functions H_1 and H_2 such that: $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_2 : \mathbb{G}_T \rightarrow \{0,1\}^l$ where l is the size in bits of the message M which is going to be ciphered. A user public key is computed as $Pub_{ID} = H_1(ID)$ and its corresponding private key is generated by PKG as $Priv_{ID} = \frac{1}{Pub_{ID}+s}P$. In order to encrypt M, the ciphering station chooses a random number $k \in \mathbb{Z}_q^*$ and executes the following steps:

1) $U = k \cdot (P_{pub} + Pub_{ID} \cdot P)$
2) $n = H_2(g^k)$
3) $V = M \oplus n$

The ciphered message is the pair $(U, V) \in G_1 \times \{0,1\}^l$.
The recipient of the previous message (U, V) computes first $n = H_2(\hat{e}(U, Priv_{ID}))$. Then, it recovers the message M as: $M = V \oplus n$.

III. IEEE 802.11s Mesh Network Architecture

The IEEE 802.11s mesh network architecture is based on the IEEE 802.11 architecture which is formed by Stations (STAs), Access Points (APs) and a Distribution System (DS) [9]. In 802.11, every AP offers connectivity to a number of STAs and forms a Basic Service Set (BSS). The DS serves to interconnect different BSSs through a wired network.
The IEEE 802.11s standard introduces modifications to the 802.11 architecture. First, the wired DS is replaced by a backbone composed of a set of wireless Mesh Points (MPs) (called also Mesh Routers or Mesh STA -MSTA). These wireless MPs provide multi-hop paths and peer to peer communications between the Mesh APs (MAPs). A MAP has the same capability as a traditional AP combined with a mesh router function. Mesh points which offer connectivity to external networks (either 802 LANs or layer 3 networks) are called Mesh Portals (MPP) or Gateways. All these components (MPs, MAPs and MPPs) form the Mesh BSS (MBSS).
The 802.11s architecture defines new functions for some mesh STAs in order to provide security services such as station authentication and key derivation. The first function is the Mesh Authenticator (MA) which acts as a pass-through server for the supplicant mesh STA by forwarding its authentication frames to the network Authentication Server (AS) [10]. In addition, the standard defines the Mesh Key Distributor (MKD) as the entity that derives the keys needed for the 4-Way Handshake that occurs between MA and the supplicant mesh STA. MKDs serve to distribute the key derivation function that was used to be performed by AS (in IEEE 802.11 networks). Each MA must be connected to an MKD because the latter is going to provide the former with the key needed to secure the communication with the supplicant.
When a mesh STA joins the network, it chooses a MA which acts as a pass-through server for its EAP message sent to AS. The supplicant STA authenticates itself to AS using a 802.1X authentication [11]. It sends EAP frames to MA encapsulated using the EAPOL protocol defined in [11]. The first EAP frame is the EAPOL-start frame and the upcoming frames represent responses to AS requests. MA uses the *Mesh EAP Message Transport Protocol* to transport the EAP frames to MKD which transfers them to AS [4]. The *Mesh EAP Message Transport Protocol* was defined to provide multi-hop EAP frame transport because EAP is a one-hop protocol. In fact, EAP frames are traditionally exchanged between the supplicant (e.g. STA) and the authenticator (e.g. an AP). Then, they are encapsulated over RADIUS or Diameter in order to be transferred to AS. The *Mesh EAP Message Transport Protocol* uses the mesh EAP encapsulation frame which is a multi-hop action frame. AS uses the reverse path to send EAP Requests to the supplicant. When the mesh STA authentication ends successfully, AS delegates the key derivation to MKD. MKD and the supplicant mesh STA create then the key hierarchy corresponding to the supplicant.
The standard assumes that there is a security association between the different authentication entities: AS-MKD, MKD-MA. In addition, another security association is established between MA and the supplicant mesh STA once the supplicant successfully authenticates to AS. Figure 2 illustrates the different entities that are used during the 802.11s station authentication and key derivation operations. Moreover, these entities do serve our authentication scheme presented in section IV-A.

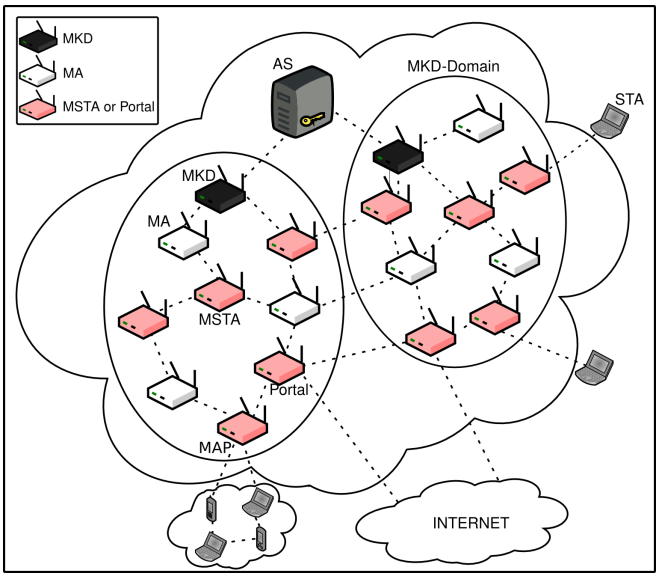

Fig. 2. IEEE 802.11s security components.

IV. ID-Based Authentication Scheme

In this section, we propose a new ID-Based authentication scheme which permits STA to authenticate itself to an AS. The authentication is based on the assumption that AS and the supplicant STA are initialized with a shared secret (i.e. a password). That is, AS and STA are using the ID-Based cryptography concepts in order to exchange this password

and to derive the keys needed to secure the exchanged messages.

When STA joins the network for the first time, it starts an authentication with AS. During this authentication, STA and AS verify alternatively their passwords. If the authentication is successful, AS orders MKD to generate STA private key as described in section IV-A. For subsequent authentications, STA uses a signature mechanism to authenticate itself to any peers.

While adapting the ID-Based cryptography concepts to the 802.11s mesh architecture, we faced the problem of the key escrow attack that could be performed by MKD. In fact, MKD is able to deduce either AS private key or any STA private key because we use it as a PKG, and by definition PKG generates every STA private key. As such, MKD is able to impersonate as AS or STA. To counteract these possible impersonation attacks, we proposed a mechanism that uses a *token*. That is, MKD only generates a partial private key for STA while STA generates the other part of its private key using a secret that is bound to the information included in the *token*. In addition, we enhanced the *public elements* with a new EC point that is only used for AS signature verification. AS secretly computes this point P_{AS} and its corresponding private key. In fact, AS does not rely on MKD for its private key computation. Furthermore, AS is in charge of defining the *public elemen ts* and distributing them over the different MKDs.

First, AS runs a Probabilistic Polynomial Time (PPT) algorithm as cited in section II. This algorithm generates the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and the bilinear mapping $\hat{e}$. AS then extends these elements with the hash functions and the two public EC points P and P_{pub} in order to get the *public elements* which are necessary for the ID-Based cryptography usage.

The point P_{AS} is computed such that $P_{AS} = s_{AS} \cdot P$ where $s_{AS} \in \mathbb{Z}_q^*$ is a secret only known by AS and it verifies $s_{AS} \neq s$. As such, every STA is able to verify an AS signature which is computed using its $Priv_{AS}$. To do so, STA has only to replace P_{Pub} by P_{AS} in BLMQ signature. The interest of introducing P_{AS} is to avoid that MKD impersonates as AS. Of course, AS entrusts MKDs for STAs partial private key derivation. That is why, AS provides MKDs with the *public elements* and the secret s used for P_{pub} computation. However, it does not provide the secret s_{AS}, nor it does not use the same secret s to compute AS private key such that $Priv_{AS} = s \cdot Pub_{AS}$. Otherwise, every MKD would be able to impersonate as AS and to recover STAs passwords during their authentications.

Note that the *public elements* are defined according to the selected IBE and IBS schemes that are going to be used between the different STAs. In our case, we consider that we are using Chen et al. encryption and BLMQ signature during the protocol execution. Consequently, the *public elements* that AS has to generate are: $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, g, P, P_{pub}, P_{AS}, H_1, H_2, H_3\}$ where: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$, $H_2 : \{0,1\}^* \times \mathbb{G}_T \rightarrow \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_T \rightarrow \{0,1\}^n$.

MKD generates a partial private key using STA public key and the secret s received from AS as $Part_{key} = \frac{1}{Pub_{STA}+s} P$. When receiving this partial private key, STA has to combine it with a secret random r to generate its full private key $Priv_{STA} = \frac{1}{r \cdot (Pub_{STA}+s)} P$. This random value was previously sent with privacy to AS. In fact, STA sends $(P_1 = r \cdot P)$ and $(P_2 = r \cdot P_{pub})$ to AS. STA then requires from AS to generate a *token* that contains P_1 and P_2, the lifetime of the private key being computed and its identity.

A. Station Initial Authentication

The initial authentication occurs when STA joins the network for the first time or after being disconnected for a while. To perform an ID-Based authentication, STA must first get the *public elements* that are published by AS. Then STA authenticates itself to AS using a preshared secret. This secret may be a password, and is noted as *pwd* in our authentication scheme.

When STA initially joins the network, it receives the Beacon frames from its one-hop neighbors. The Beacon frame contains the *Mesh Security Capability* information element [4]. This information element indicates whether the sender of the Beacon is an MA. In addition, it indicates the *MKD domain* to which this MA is connected. Based on the configuration information carried in the received Beacon, the newly arrived STA selects MA which is going to relay its authentication frames. After choosing its MA, STA starts the authentication scheme presented in Figure 3.

• **Message 1:** this message is sent to AS. It is referenced as the *Start-authentication* message. This first message like all the subsequent authentication messages transits through MA. The supplicant STA includes a nonce n_1 and its identity (ID_{STA}) in this message. The nonce n_1 is chosen randomly to prove the freshness of the message. It is used to prevent from replay attacks. The identity ID_{STA} represents the identity which is going to be used for STA public key derivation.

In order to avoid a DoS attack on AS, we suppose that MA is only accepting a certain number of requests T_0 coming from the same STA during a certain period of time. In addition, AS does not accept more than T_1 authentication requests coming from the same MA.

When AS receives this message 1, it looks for STA *pwd* in its password database using the received identity ID_{STA}. AS uses this *pwd* for generating message 2.

• **Message 2:** it is generated by AS as a response to message 1. It contains a new nonce n_2, the received nonce n_1, the current *public elements* and an AS signature. AS signature is computed over the string formed by the concatenation of n_1, n_2, the identities of AS, STA and MKD (ID_{AS}, ID_{STA} and ID_{MKD}), the *public elements* (PE) and STA *pwd*. AS includes in purpose the identity of MKD, that transmitted the message 1 from STA, to make STA identify its future partial private key generator (namely this MKD). Meanwhile, the *pwd* is included only in the hash which is signed by $Priv_{AS}$. In our case BLMQ

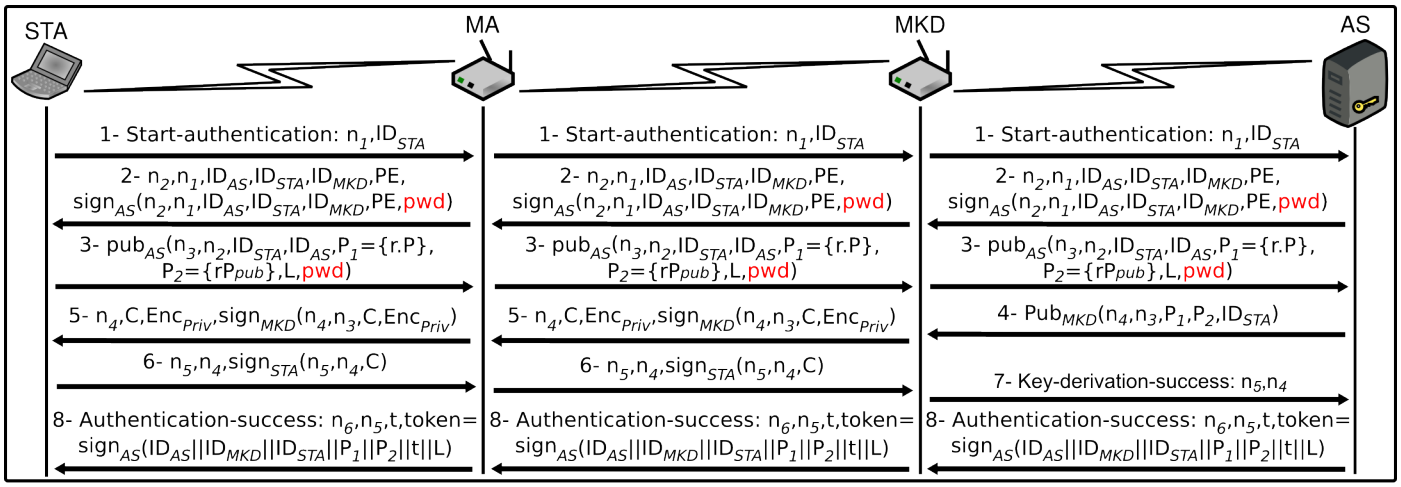

Fig. 3. STA initial authentication scheme.

signature is used. So AS computes the pair $(S,h) = ((k+h)Priv_{AS}, H_2(n_1||n_2||ID_{AS}||ID_{STA}||ID_{MKD}||PE||pwd,g^k))$ where $k \in \mathbb{Z}_q^*$.

When the supplicant STA receives message 2, it verifies its freshness by checking the values of n_1 and n_2. Then it uses the *public elements* to derive AS's public key. The public key derivation consists in hashing the identity of AS using H_1: $Pub_{AS} = H_1(ID_{AS})$. STA concatenates its *pwd* to n_1, n_2, ID_{AS}, ID_{STA}, ID_{MKD} and the *public elements* (PE) before checking the validity of AS signature. In practice STA checks the equality between h and $H_2(n_1||n_2||ID_{AS}||ID_{STA}||ID_{MKD}||PE||pwd,\hat{e}(S, Pub_{AS}+P_{AS})g^{-h})$. If the verification is successful, STA authenticates AS and the *public elements*. Else, STA stops the authentication processing.

• **Message 3:** the supplicant STA chooses two random numbers n_3 and r. The random value n_3 is sent to AS. It will be used during the key encoding by MKD. However, the random value r is kept secret by STA because it will be used for STA private key computation with the partial key received from MKD. The supplicant STA then computes $(P_1 = r \cdot P)$ and $(P_2 = r \cdot P_{pub})$ before generating message 3.

Message 3 contains the nonces n_3 and n_2, the points P_1 and P_2, the *pwd* and a validity period which represents STA proposed lifetime (L) for its upcoming private key. All these fields are ciphered using AS public key Pub_{AS}. As we use Chen et al. encryption, the message 3 contains the pair: $(U,V) = (k \cdot (P_{AS} + Pub_{AS} \cdot P), (n_3||n_2||ID_{AS}||ID_{STA}||P_1||P_2||L||pwd) \oplus H_3(g^k))$ where $k \in \mathbb{Z}_q^*$. When receiving this message, AS authenticates STA using the *pwd*. In addition, it verifies that the points P_1 and P_2 are computed using the same random r by verifying the equality: $\hat{e}(P,P_2) = \hat{e}(P_1, P_{pub})$. This verification checks also that STA computed P_1 and P_2 with respect to the *public elements* received in message 2.

• **Message 4:** after successfully authenticating STA, AS sends a new nonce n_4, the points P_1 and P_2, and STA identity ID_{STA} to MKD for partial private key generation. AS adds to the previous list of element to the received nonce n_3. These elements are sent encrypted using Pub_{MKD}. That is, AS sends the pair $(U,V) = (k \cdot (P_{pub} + Pub_{MKD} \cdot P), (n_3||n_4||ID_{STA}||P_1||P_2) \oplus H_3(g^k))$ where $k \in \mathbb{Z}_q^*$.

Upon receiving message 4 from AS, MKD stores n_4 for the upcoming message (message 5). Note that the same sequence number is going to be used in message 5 to easily identify request and response messages between the supplicant STA and AS. In addition, MKD generates STA partial private key as $Part_{key} = \frac{1}{Pub_{STA}+s}P$.

• **Message 5:** after generating STA partial private key, MKD encodes it as $Enc_{Priv} = Part_{key} + (n_3 \cdot P_{pub})$. Recovering $Part_{key}$ from the encoded private key is equivalent to solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) [12]. The encoded partial key is then signed by MKD before being sent in message 5 to STA. Message 5 contains also a challenge C and the same nonce as in message 4: $n4$. The challenge will be used to prove that STA recovery of its private key is successful.

When receiving message 5, STA which knows n_3 and P_{pub} recovers its partial private key, as follows: $Part_{key} = Part_{key} + (n_3 \cdot P_{pub}) - (n_3 \cdot P_{pub})$. Consequently, the supplicant STA becomes able to compute its full private key as $Priv_{STA} = r^{-1} \cdot Part_{key} = \frac{1}{r \cdot (Pub_{STA}+s)}P$.

• **Message 6:** at this point, STA signs of the challenge C with its $Priv_{STA}$ using the modified BLMQ signature which is presented in section IV-B. This message contains in addition to the received random n_4 a new random n_5. Upon receiving message 6, MKD verifies STA signature of the challenge C. If the signature is valid, MKD deduces that the supplicant STA has successfully derived its private key. As a consequence, MKD requests from AS to generate

STA's *token* in **message 7**.

• **Message 8:** it is sent by AS to complete the authentication and to deliver the *token* to STA. It is called the *Authentication-success* message. The *token* includes STA identity, AS identity, MKD identity, the points P_1 and P_2, the lifetime of STA private key (L) and a timestamp t: $(token{=}Sign_{Priv_{AS}}\{ID_{AS}||ID_{MKD}||ID_{STA}||t||L\})$.
After getting its *token*, STA can start communications with its peers. It can use the modified BLMQ signature with its *token* to authenticate itself with any peer. The modified BLMQ signature and the modified Chen et al. encryption algorithm that have been adapted to the presence of the *token* as detailed in the section IV-B.

B. Signature Adaptation to STA Private Key Generation

After its authentication to AS, STA uses a signature mechanism along with a *token* to authenticate itself to its peer STAs. That is, we modified BLMQ signature so that it becomes tightly related to the *token* and especially to the value of P_1, P_2 and the secret r. As such, the signature sent by STA makes it possible for the peer STAs to verify that STA owns the private key bound to the points P_1 and P_2 given within the *token*.
When STA A wants to authenticate STA B, A sends a challenge to B encrypted with Pub_B which is deduced from the identity of B. Then, B must respond with a message containing its signature of the challenge and a copy of its *token*. The signature validity depends on the values of P_{1B} and P_{2B} included in the *token*. In fact, the signature modification concerns the use of the secret value r by the signer and the use of P_1 and P_2 by the verifier. So if the signature is valid, the verifier deduces that the signer knows the true value of the secret r and it was successfully authenticated by AS.
In the following we present the modifications that we introduced to BLMQ signature in order to include the use of STA secret element r. First, STA generates its private key as $Priv_{STA} = \frac{1}{r\cdot(Pub_{STA}+s)}P$. Then, it chooses a random $k \in \mathbb{Z}_q^*$ and computes the signature as:

1) $n = g^k$
2) $h = H_2(M, n)$
3) $S = (k + h)Priv_{STA}$

The signature is the pair $(S, h) \in G_1 \times \mathbb{Z}_q^*$. Then, the signature verifier has only to check the equality between h and $H_2(M, \hat{e}(S, H_1(ID)P_1 + P_2)g^{-h})$ where P_1 and P_2 are taken from signer's *token*. This signature verification holds because:
$h' = H_2(M, \hat{e}(S, H_1(ID)P_1 + P_2)g^{-h})$
$\implies h' = H_2(M, \hat{e}((k + h)Priv_{STA}, H_1(ID) \cdot r \cdot P + r \cdot P_{pub})g^{-h})$
$\implies h' = H_2(M, \hat{e}(\frac{k+h}{r\cdot(Pub_{STA}+s)}P, H_1(ID)\cdot r\cdot P + r\cdot s\cdot P)g^{-h})$
$\implies h' = H_2(M, \hat{e}((k + h) \cdot P, P)g^{-h})$
$\implies h' = H_2(M, g^{h+k} \cdot g^{-h})$
$\implies h' = h$
We adapted also Chen et al. encryption to include the points P_1 and P_2. In order to encrypt a message M, the ciphering station chooses a random number $k \in \mathbb{Z}_q^*$ and executes the following steps:

1) $U = k \cdot (P_2 + Pub_{ID} \cdot P_1) = k \cdot r \cdot (s + Pub_{ID}) \cdot P$
2) $n = H_2(g^k)$
3) $V = M \oplus n$

The ciphered message is the pair $(U, V) \in G_1 \times \{0, 1\}^l$.
The recipient of the previous message (U, V) computes first $n = H_2(\hat{e}(U, Priv_{ID}))$. Then, it recovers the message M as: $M = V \oplus n$. The decryption holds because:
$g^{k'} = \hat{e}(U, Priv_{ID})$
$\implies g^{k'} = \hat{e}(k \cdot r \cdot (s + Pub_{ID}) \cdot P, \frac{1}{r\cdot(Pub_{STA}+s)}P)$
$\implies g^{k'} = \hat{e}(k \cdot P, P)$
$\implies g^{k'} = \hat{e}(P, P)^k$
$\implies g^{k'} = g^k$
In reality, any ID-Based signature or encryption algorithm can be changed to take into consideration the presence of the *token*. In fact, these modification can be applied to any scheme that relies on a pairing function $\hat{e}$. We only have to add the secret r during the signature generation or message decryption. In addition, we have to modify the signature verification algorithm in order to take into consideration the point $r \cdot P$ (and if necessary the point $r \cdot P_{pub}$). The point is that r and r^{-1} disappear when multiplication is used in $\mathbb{G}_T$.

TABLE I
IBS AND IBE ELEMENTARY OPERATIONS.

IBS or IBE scheme	$\mathbb{G}_T$ Exp	Point/scalar mul	Pairings
Paterson signature	0	4	0
Paterson verification	2	0	3
Hess signature	1	2	1
Hess verification	1	0	2
BLMQ signature	1	1	0
BLMQ verification	1	1	1
BF encryption	1	1	1
BF decryption	0	0	1
BB encryption	1	3	0
BB decryption	0	0	2
Chen et al. encryption	1	1	0
Chen et al. decryption	0	0	1

C. Reasons for Selecting Sakai-Kasahara Key Construction Scheme

To compare the performances of IBS or IBE schemes, a first analysis of the number of mathematical operations can be done. Likely to Barreto et al. [8], the signature scheme performances can be compared according to the number of $\mathbb{G}_T$ exponentiations, scalar point multiplications and pairing computation operations. Table I establishes such a comparison between Paterson signature [13], BLMQ signature and Hess signature [14]. In addition, it presents a comparison of the following encryption schemes: Boneh and Franklin (BF) [7], Boneh and Boyen (BB) [15] and Chen et al [5].
According to Table I, BLMQ signature and Chen et al. encryption are expected to be more efficient than the other signature and encryption schemes as they rely on only one

TABLE II
EQUIVALENT KEY SIZES FOR THE SAME SECURITY LEVEL (IN BITS).

k	RSA key length	ECC key length
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

pairing computation. This gain in computation time results from using the Sakai-Kasahara key construction scheme in BLMQ signature and Chen et al. encryption. Meanwhile, the other aforementioned signature and encryption algorithms rely on a more classical key construction where the public key is computed as: $Pub_{ID} = H(ID)$ while the private key is calculated as: $Priv_{ID} = s \cdot Pub_{ID}$. That is, the public and private keys are two points of an elliptic curve. However, when Sakai-Kasahara key construction is used, the public key is a scalar. So, with BLMQ and Chen et al. encryption, we do not only avoid the hashing into a point to compute the public key, but also we reduce the number of pairing computations during the signature generation and verification (respectively encryption and decryption).

In cryptography, the security level of a symmetric encryption algorithm is defined as the number of operations needed to break the algorithm when a k-bit key is used. For example, the number of elementary operations needed to break a block cipher encryption scheme is equal to 2^k [16]. In asymmetric cryptography, the security level of an algorithm is defined with respect to the hardness of solving the Discrete Logarithm Problem (DLP) either in a multiplicative group (the case of RSA) or an additive group (the case of ECDSA). This concept of security level sets the length in bits of RSA keys and EC keys. For example, Table II presents the equivalence between the lengths of RSA and EC keys respectively to the security level k, where k corresponds to the security level of a k-bit length symmetric key.

The security level of an ID-Based cryptographic scheme depends on the security level of the pairing function in use $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. However, the security level of $\hat{e}$ is related to the hardness of solving the DLP in the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and as such is closely related to the groups being selected as some of them make the DLP easier. To understand how to define this security level in practice, investigation of the structures of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ is necessary [2].

Table III confirms the aforementioned theoretical results through testing. We used the PBC library to implement the previous signature and encryption schemes [17] and compare them at the same security level. 1000 samples serve to evaluate the signature generation and verification times. All the tests were performed on an Intel(R) Core 2 Duo machine running at 800 MHz each.

TABLE III
SIGNATURE GENERATION AND VERIFICATION TIMES (IN S).

Security level	80	112	128
Paterson	0.017316	0.059288	0.123116
Hess	0.024889	0.091835	0.204757
BLMQ	0.007050	0.023145	0.047907
Signature verification time			
Paterson	0.024135	0.106804	0.264832
Hess	0.013232	0.060985	0.155714
BLMQ	0.012653	0.051669	0.121929
Encryption time			
BF	0.013157	0.051919	0.122760
BB	0.018031	0.062644	0.129770
Chen et al.	0.007044	0.023128	0.047845
Decryption time			
BF	0.006937	0.029730	0.075844
BB	0.013020	0.058707	0.149972
Chen et al.	0.006959	0.030095	0.075357

D. Security Discussion

In this section, we present informally how the aforementioned authentication protocol removes some attacks and how it can be enhanced to limit other threats. We use an 'active saboteur' attacker model as defined by Dolev and Yao in [18]. That is, an attacker might be a user of the network and can have access to all the traffic.

• *Denial of service attack (DoS):* To avoid that an attacker makes a DoS attack against AS by sending a big amount of *Start-authentication* messages, we decided to limit the number of accepted authentication requests to a threshold T_0 by the MA and to a threshold T_1 by the AS. As such, MA is only accepting T_0 authentication requests coming from the same supplicant STA during a certain period of time, and AS is only accepting T_1 authentication requests from the same MA. When the number of authentication requests exceeds T_0 or T_1, MA or AS drops all the upcoming packets received from the supplicant STA or MA respectively.

The use of T_0 at MA removes DoS attacks but does not help against Distributed DoS attack (DDoS). An attacker controling many STAs can launch a DDoS attack against AS by flooding AS with many *Start-authentication* messages originating from different (zombie) STAs under control. AS flooding is precluded thanks to the second threshold T_1.

• *Replay attack:* For avoiding replay attacks, we make use of nonces (n_1,... and n_6), and we assume that these random numbers are at least 128 bit length. As a consequence of which, the probability for getting the same random number for two consecutive authentication sessions is equal to $1/2^{128}$.

Nonces enable introducing a strong ordering and association between messages as all the messages include their own nonce and the nonce of the previous message. As such, an attacker is able to impersonate STA or AS, however he will not be able to replay old messages unless nonces of an older authentication session match the ones of the current session. For example, an attacker impersonating STA, has to replay an old message 3. However, for the replay to be successful, the value n_2 in message 3 must match the same value n_2 included in message 2. The probability for getting the same

value n_2 is equal to $1/2^{128}$ in case only one message 3 is known for replay by the attacker, but it increases in case the attacker knows more than one valid message 3.
Using random numbers to counteract replay attacks assume that STA and AS store the list of nonces already used and check the nonce of each received message against this list. To get rid of the storage issue, nonces can be replaced with timestamps, but other issues like STAs synchronization during timestamps verification then arise.

• *Private key recovery by an attacker*: Concerning the private key recovery, an attacker needs first to recover the partial private key of STA. He may get the encoded private key of a supplicant but he has to find the secret nonce n_3 in order to recover the partial private key of the supplicant. The problem of finding n_3 is equivalent to the discrete logarithm problem over an elliptic curve group (ECDLP) [12]. In addition, the full private recovery requires from an attacker to guess r from P_1 or P_2 which comes also to solve the ECDLP.

• *Key escrow attack*: For counteracting the key escrow problem, every STA participates to its private key generation with a secret value r. This secret is included in the public *token* but after being multiplied by the points P and P_{pub}, so neither AS nor MKD are able to compute the private key corresponding to (P_1 and P_2). Note that AS can technically generate a fake *token* with a fake ($P'_1 = r' \cdot P$) and ($P'_2 = r' \cdot P_{pub}$) in order to impersonate as STA, however, this is in contradiction with the assumption that AS must be trustworthy.

V. Conclusion

In this paper, we present an ID-Based authentication scheme for a mesh network station STA to authenticate and initialize its key pair while being protected against the key escrow attack. To closely match the IEEE 802.11s mesh network needs, we adapted our authentication scheme to the IEEE 802.11s mesh architecture by assigning a specific role to the Mesh Key Distributor (MKD). In addition, we make use of the Sakai-Kasahara cryptographic schemes to achieve high performance as they are known as the fastest ID-Based cryptographic schemes at the moment.

References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 47–53.

[2] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels, *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. New York, NY, USA: Cambridge University Press, 2005.

[3] K. G. Paterson and G. Price, "A comparison between traditional public key infrastructures and identity-based cryptography," Royal Holloway University of London, Tech. Rep., 2003.

[4] *IEEE P802.11s/D2.06: Part 11: Wireless LAN MAC and Physical layer specifications. Amendment 10: Mesh networking*, IEEE Working Draft Proposed Standard, Rev. 2.06, jan 2009.

[5] L. Chen, Z. Cheng, J. Malone-Lee, and N. Smart, "Efficient id-kem based on the sakai-kasahara key construction," vol. 153, pp. 19–26, 2006.

[6] J. Baek, J. Newmarch, R. Safavi-naini, and W. Susilo, "A survey of identity-based cryptography," in *Proc. of Australian Unix Users Group Annual Conference*, 2004, pp. 95–102.

[7] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 2001, pp. 213–229.

[8] P. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater, "Efficient and provably-secure identity-based signatures and signcryption from bilinear maps," in *Advances in Cryptology - ASIACRYPT 2005*, ser. Lecture Notes in Computer Science, B. Roy, Ed., vol. 3788. Springer Berlin / Heidelberg, 2005, pp. 515–532.

[9] *IEEE Std 802.11-2007: Part 11: Wireless LAN MAC and Physical layer specifications*, IEEE Standard, Rev. Revision of IEEE Std 802.11-1999, jun 2007.

[10] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748 (Proposed Standard), Internet Engineering Task Force, Jun. 2004, updated by RFC 5247. [Online]. Available: http://www.ietf.org/rfc/rfc3748.txt

[11] *IEEE Std 802.1X-2004: Port Based Network Access Control*, IEEE Standard, dec 2004.

[12] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[13] K. G. Paterson, "Id-based signatures from pairings on elliptic curves," *Electronics Letters*, vol. 38, no. 18, pp. 1025 – 1026, 29 2002.

[14] F. Hess, "Efficient identity based signature schemes based on pairings," in *SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*. London, UK: Springer-Verlag, 2003, pp. 310–324.

[15] D. Boneh and X. Boyen, "Efficient selective-id secure identity-based encryption without random oracles," in *Advances in Cryptology - EUROCRYPT 2004*, vol. 3027. Springer Berlin-Heidelberg, 2004, pp. 223–238.

[16] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, pp. 3113 – 3121, 2008, applications of Algebra to Cryptography. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X08000449

[17] B. Lynn, "On the implementation of pairing-based cryptosystems," Ph.D. dissertation, STANFORD UNIVERSITY, 2007.

[18] D. Dolev and A. Yao, "On the security of public key protocols," *Information Theory, IEEE Transactions on*, vol. 29, no. 2, pp. 198 – 208, mar 1983.