Key-escrow Resistant ID-based Authentication Scheme for IEEE 802.11s Mesh Networks

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, ID-based cryptography is reported as an alternative to Public Key Infrastructures (PKI). It proposes to derive the public key from the node's identity directly. As such, there is no need for public key certificates, and direct benefit of this is to remove the burdensome management of certificates. However, the drawback is the need for a Private Key Generator (PKG) entity which can perform a key escrow attack. In this article, we present an ID-based authentication scheme that is adapted to the IEEE 802.11s mesh networks and resistant against key escrow attacks.**

I. Introduction

ID-based cryptography was initially introduced by A. Shamir [1] to provide entities with public/private key pairs with no need for certificates, Certification Authority (CA) and PKI. Shamir assumes that each entity uses a pair of its identifiers as its public key. These identifiers have to be unique. 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.

During the last decade, the ID-based cryptography has been enhanced by 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 the 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 the 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 the 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 the latter's ciphered traffic.

In this article, we present a new ID-based authentication mechanism for IEEE 802.11s mesh networks. The main idea is to authenticate every station by an Authentication Server (AS) which delegates the station key generation to the PKG. We propose to make the PKG generate a partial private key for every station. Then, the concerned station uses this partial private key as a base to compute its own private key. In the following, we show how this partial private key generation avoids the key escrow threat of the PKG.

The article is organised as follows. After presenting the ID-based cryptography, we introduce the IEEE 802.11s mesh architecture. Then we present our authentication scheme followed by its security analysis and its possible enhancements. Finally conclusions are given.

II. ID-based Key Generation and Signatures

When a node requires a private key, it provides the PKG with the identity ID that it intends to use for its private key computation. The PKG then derives the node's private key using some parameters which must be defined in respect to the Bilinear Diffie-Hellman problem [5]. In order to generate these parameters, the PKG runs a Probabilistic Polynomial Time (PPT) algorithm which takes as input a security parameter k and outputs the groups G_1 and G_2 and the pairing function $\hat{e}$ from $G_1 \times G_1$ in G_2. G_1 is an additive group of prime order q and G_2 is a multiplicative group of the same order q. Generally, G_1 is a subgroup of the group of points of an Elliptic Curve (EC) over a finite field and G_2 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 there exists a point $P \in G_1$ such that $\hat{e}(P,P) \neq 1_{G_2}$. The point P is used to compute another point P_{pub}. Practically this kind of bilinear mapping is derived from the Weil or Tate pairing [6].

In addition to the groups definition, some hash functions need to be defined in accordance to the ID-based signature or encryption schemes that are going to be used. For example a hash function H that verifies $H : \{0,1\}^* \rightarrow G_1$ is defined in order to transform the node's identity into an

EC point. The list containing the groups G_1 and G_2, the bilinear mapping $\hat{e}$, the points P and P_{pub} and the hash functions forms the *public elements*. These *public elements* are distributed by the PKG to the network users because they are required during the public key derivation and the signature operation.

The key derivation operation starts when the PKG receives the ID of the node that is requesting a private key. First, the PKG computes the user's public key as $Pub_{ID} = H(ID)$. Then, the PKG generates the corresponding private key using a local secret value $s \in \mathbb{Z}_q^*$. Note that the private key is computed as: $Priv_{ID} = s \cdot Pub_{ID}$. The secret value s is also used for P_{pub} derivation from P: $P_{pub} = s \cdot P$. It is clear from the aforementioned key derivation scheme that the PKG is going to know every private key that it generates. This supposes that the PKG is a trustworthy entity which is not going to impersonate as the private key owner by generating signature or deciphering encrypted traffic aimed to the legitimate station. Generally, stations must ensure that the PKG is not going to make a key escrow attack.

A. Paterson Signature Scheme

K.G. Paterson proposed in 2002 an ID-based signature scheme using ECC [7]. He defines three hash functions H_1, H_2 and H_3 such that: $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_3 : G_1 \rightarrow \mathbb{Z}_q^*$. So, Paterson *public elements* are $\{G_1, G_2, q, \hat{e}, P, P_{pub}, H_1, H_2, H_3\}$. The PKG computes the user's public key as $Pub_{ID} = H_1(ID)$. Then, the PKG generates the corresponding private key using a local secret value $s \in \mathbb{Z}_q^*$.

In order to compute the signature of a message M, a user generates a secret random $k \in \mathbb{Z}_q^*$ and computes its signature as the pair $(R, S) \in G_1 \times G_1$ where: $R = k \cdot P$, $S = k^{-1}(H_2(M) \cdot P + H_3(R) \cdot Priv_{ID})$.

The signature verifier has only to compare $\hat{e}(R, S)$ to $\hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{ID})^{H_3(R)}$. The two values must be equal in order to consider the signature as valid.

B. Hess Signature Scheme

F. Hess presented its ID-based signature scheme in 2003 [8]. His signature scheme keeps the previous public parameters definition but it replaces H_2 and H_3 by a new hash function that we denote as $H_4 : \{0,1\}^* \times G_2 \rightarrow \mathbb{Z}_q^*$. In order to sign a message M, the user chooses an arbitrary point $P_1 \in G_1^*$ and a random number $k \in \mathbb{Z}_q^*$. In addition, it executes the following steps:

1) $r = \hat{e}(P_1, P)^k$
2) $v = H_4(M, r)$
3) $U = v \cdot Priv_{ID} + k \cdot P_1$

The signature is formed by the pair $(U, v) \in G_1 \times \mathbb{Z}_q^*$.

The signature verifier then has to compute:

1) $r = \hat{e}(U, P) \cdot \hat{e}(Pub_{ID}, -P_{pub})^v$
2) The signature is accepted if and only if $v = H_4(M, r)$

Integration of this ID-based signature scheme into an EAP authentication method was submitted by Wenju et al. in 2009 [9].

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) [10]. 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). The functions of "supplicant", "authenticator" and "authentication server" are inherited from the EAP standard [11]. 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 the MA and the supplicant mesh STA. The MKDs serve to distribute the key derivation function that was used to be performed by the 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 will act as a pass-through server for its EAP message sent to the AS. The supplicant STA authenticates itself with the AS using a 802.1X authentication [12]. It sends EAP frames to the MA encapsulated using the EAPOL protocol defined in [12]. The first EAP frame is the EAPOL-start frame and the upcoming frames represent responses to the AS requests. The MA uses the *Mesh EAP Message Transport Protocol* to transport the EAP frames to the MKD which transfers them to the 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. a STA) and the authenticator (e.g. an AP). Then, they are encapsulated over RADIUS or Diameter in order to be transferred to the AS. The *Mesh EAP Message Transport Protocol* uses the mesh EAP encapsulation frame which is a multi-hop action frame.

The AS uses the reverse path to send EAP Requests to

the supplicant. When the mesh STA authentication ends successfully, the AS delegates the key derivation to the MKD. The MKD and the supplicant mesh STA create 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 created between the MA and the supplicant mesh STA after a successful authentication of the latter by the AS. Figure 1 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.

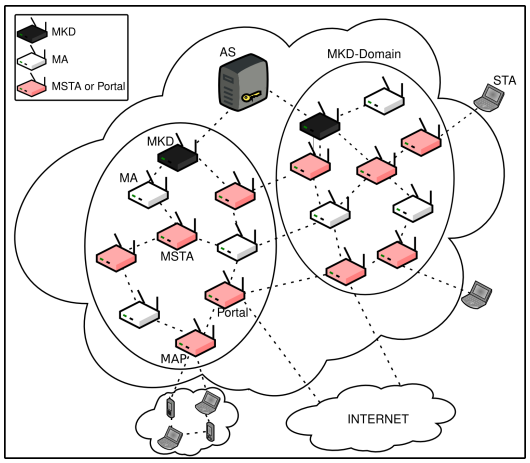

Fig. 1. IEEE 802.11s security components.

IV. ID-based Authentication Scheme

Nowadays, most of the authentication schemes that are being proposed for wireless networks uses the 802.1X standard [12]. The authentication method proposed by this standard are either based on the verification of a secret shared between two STAs or a signature mechanism that uses certificates to authenticate the public/private key pair used for signing. Management of public/private key requires deploying CAs to control the generation, revocation and duration of certificates. This is disadvantageous in wireless environments, such as ad-hoc and mesh networks, where stations may have some power and memory constraints and CA reachability is not guaranteed.
To mitigate these disadvantages while keeping usage of public/private key, we propose a new ID-based authentication scheme which permits STA to authenticate itself to an AS when it initially joins the network. The authentication is based on the assumption that the AS and the supplicant STA are initialized with a shared secret (i.e. a password). That is, the AS and the 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 (or MSTA) joins the network for the first time, it starts an authentication with the AS. During this authentication, the STA and the AS verify alternatively their passwords. If the authentication is successful, the AS orders the MKD to generate the STA private key as described in section IV-A. For subsequent authentications, the 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 the MKD. In fact, the MKD is able to deduce either the AS private key or any STA private key because we use it as a PKG, and by definition the PKG generates every STA private key. As such, the MKD is able to impersonate as the AS or the STA. To counteract these possible impersonation attacks, we proposed a mechanism that uses a *token*. That is, the MKD only generates a partial private key for the STA while the 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 the AS signature verification. The AS secretly computes this point P_{AS} and its corresponding private key. In fact, the AS does not rely on the MKD for its private key computation. Furthermore, the AS is in charge of defining the *public elements* and distributing them over the different MKDs.
First, the AS runs a Probabilistic Polynomial Time (PPT) algorithm as cited in section II. This algorithm generates the groups G_1 and G_2 and the bilinear mapping $\hat{e}$. The 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, the STA has only to replace P_{Pub} by P_{AS} in Paterson or Hess signature schemes. The interest of introducing P_{AS} is to avoid that the MKD impersonates as the AS. Of course, the AS entrusts the MKDs for the STAs partial private key derivation. That is why, the 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 the AS private key such that $Priv_{AS} = s \cdot Pub_{AS}$. Otherwise, every MKD would be able to impersonate as the AS and to recover the STAs passwords during their authentications.
The MKD generates a partial private key using the STA public key and the secret s received from the AS. When receiving this partial private key, the STA has to combine it with a secret random r to generate its full private key $Priv_{STA}$. This random value was previously sent with privacy to AS. In fact, the STA sends $(r \cdot P)$ to the AS. The STA then requires from the AS to generate a *token* that contains $(r \cdot P)$, 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, the STA must first get the *public elements* that are published by the AS. Then STA authenticates itself to the AS using a preshared secret. This secret may be a password, and is noted as *pwd* in our authentication scheme.

Note that the *public elements* are defined according to the selected ID-based signature scheme. If Paterson signature scheme is in use, the *public elements* are: $\{G_1, G_2, \hat{e}, H_1, H_2, H_3, P, P_{pub}, P_{AS}\}$. If Hess signature scheme is used, the *public elements* are: $\{G_1, G_2, \hat{e}, H_1, H_4, P, P_{pub}, P_{AS}\}$. These parameters are used for signature or keys computations.

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 the MA which is going to relay its authentication frames. The Beacon frame gives the STA information about its future MA clock so that the STA synchronizes its clock according to the MA's one. Clock synchronization is important in order to get accurate timestamps. After choosing its MA, the STA starts the authentication scheme presented in Figure 2.

- **Message 1:** this message is sent to the AS. This

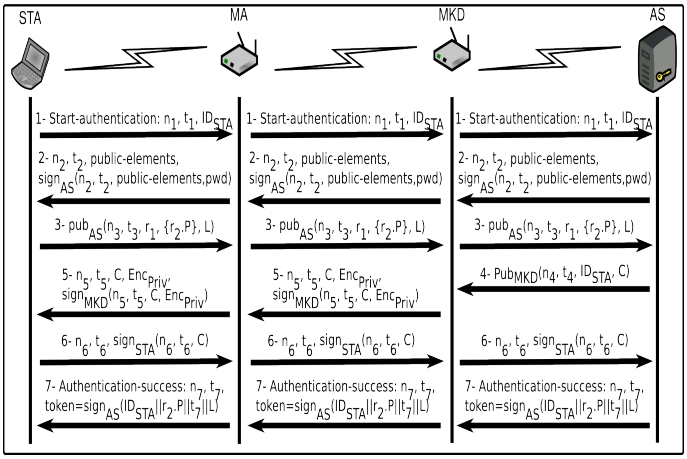

Fig. 2. ID-based authentication scheme.

message is referenced as the *Start-authentication* message. This first message like all the subsequent authentication messages transits through the MA.

The supplicant STA includes a nonce n_1, a timestamp t_1 and its identity (ID_{STA}) in this message. The n_1 is chosen randomly to prove the freshness of the message especially when it is combined with the timestamp t_1. These two values are used to prevent from replay attacks. The identity ID_{STA} represents the identity which is going to be used for the STA public key derivation.

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

When the AS receives this message 1, it checks the timestamp t_1 value in order to verify the message freshness. Then it looks for the STA *pwd* in its password database using the received identity ID_{STA}. The AS uses this *pwd* for generating message 2.

- **Message 2:** it is generated by the AS as a response to message 1. It contains a sequence number n_2 such that $n_2 = n_1 + 1$, a timestamp t_2, the current *public elements* and an AS signature. The AS signature is computed over the string formed by the concatenation of n_2, t_2, the *public elements* and the STA *pwd*. The *pwd* is included in the signature only.

When the supplicant STA receives message 2, it verifies its freshness by checking the value of t_2. In addition, it verifies the value of n_2 because it is used to bind message 2 to message 1. Then it uses the *public elements* to derive the AS's public key Pub_{AS}. The public key derivation consists in hashing the identity of the AS using H_1. The STA concatenates its *pwd* to n_2, t_2 and the *public elements* before checking the validity of the AS signature. If the signature verification is successful, the STA authenticates the AS and the *public elements*. Else, the STA stops the authentication processing.

- **Message 3:** the supplicant STA chooses two random numbers r_1 and r_2. The random value r_1 will be sent to the AS. It will be used during the key encoding by the MKD. However, the random value r_2 is kept secret by the STA because it will be used for the STA private key computation in conjunction with the partial key received from the MKD.

The supplicant STA then computes $(r_2 \cdot P)$ and generates message 3. Message 3 contains the sequence number $n_3 = n_1 + 2$, the timestamp t_3, the random r_1, the point $(r_2 \cdot P)$, the *pwd* and a validity period which represents the STA proposed lifetime (L) for its upcoming private key. All these fields are ciphered using the AS public key Pub_{AS}. When receiving this message, the AS authenticates the STA using the *pwd*.

- **Message 4:** after successfully authenticating STA, the AS sends r_1 and a challenge C to the MKD for partial private key generation. These elements are sent encrypted using Pub_{MKD}. This MKD can be identified thanks to the identity of the MA which is attached to it. Each MA connects only to one MKD.

When receiving message 4 from the AS, the MKD verifies that the timestamp t_4 is valid and it stores the sequence number $n4 = n_1 + 3$ 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 the AS. In addition, the MKD generates the STA partial private key as $Part\_Priv_{STA} = s \cdot Pub_{STA}$.

- **Message 5:** after generating the STA partial private

key, the MKD encodes it as $Enc_{Priv} = Part\_Priv_{STA} + (r_1 \cdot s \cdot P)$. Recovering $Part\_Priv_{STA}$ from the encoded private key is equivalent to solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) [2]. The encoded partial key is then signed by the MKD before being sent in message 5 to the STA. Message 5 contains also a timestamp t_5 and the same sequence number as in message 4 ($n5 = n4 = n_1+3$). In addition, it contains the challenge C. This challenge will be used to prove that the STA recovery of its private key is successful.

When receiving message 5, the STA which knows r_1 and P_{pub} recovers its partial private key, as follows: $Part\_Priv_{STA} = Part\_Priv_{STA} + (r_1 \cdot s \cdot P) - (r_1 \cdot P_{pub})$. Consequently, the supplicant STA becomes able to compute its full private key as $Priv_{STA} = r_2^{-1} \cdot Part\_Priv_{STA} = r_2^{-1} \cdot s \cdot Pub_{STA}$.

• **Message 6:** at this point, the STA requests a *token* from the AS. The STA includes the signature of the challenge C with its $Priv_{STA}$ using one of the signature schemes that are presented in the section IV-B. This message contains also the sequence number $n_6 = n_1 + 4$ and a timestamp t_6. Upon receiving message 6, the AS checks the values of the sequence number and the timestamp. In addition, it verifies the STA signature of the challenge. If the signature is valid, the AS deduces that the supplicant STA has successfully derived its private key. As a consequence, the AS generates the STA *token*. The elements that will be part of the *token* include the STA identity, the point $(r_2 \cdot P)$, the lifetime of the STA private key (L) and the AS timestamp t_7. ($token$=$Sign_{Priv_{AS}}\{ID_{STA} \| r_2 \cdot P \| t_7 \| L\}$).

• **Message 7:** it is sent by the AS to complete the authentication and to deliver the *token* to the STA. It is called the *Authentication-success* message. It contains the sequence number $n_7 = n_1 + 5$, the timestamp t_7 and the signed *token*.

After getting its *token*, the STA starts communicating with its peers. It can use one of the modified signature schemes with its *token* to authenticate itself with any peer. The modified signature schemes are Paterson and Hess signature schemes that have been adapted to the presence of the *token* as presented in the section IV-B.

B. Signature Modification

After its authentication to the AS, STA uses a signature mechanism along with a *token* to authenticate itself to its peer STAs. That is, we modified some of the existing signature schemes so they are tightly related to the *token* and especially to the value of $(r_2 \cdot P)$ and the secret r_2. As such, the signature sent by STA makes it possible for the peer STAs to verify that the STA owns the private key bound to the value $(r_2 \cdot P)$ 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 value of $(r_2 \cdot P)$ included in the *token*. In fact, the signature modification concerns the use of the secret value r_2 by the signer and the use of $(r_2 \cdot P)$ by the verifier. So if the signature is valid, the verifier deduces that the signer knows the true value of the secret r_2 and it was successfully authenticated by the AS.

In the following we present the modifications that we introduced to the different signature schemes in order to include the use of the STA secret element r_2:

• In Paterson signature scheme, a user computes the signature of a message M as the pair $(R, S) \in G_1 \times G_1$ where: $R = r_2 \cdot P, S = r_2^{-1} \cdot H_2(M) \cdot P + H_3(R) \cdot Priv_{ID}$. But only the S is sent to the verifier because the latter gets the R value from the *token*. The verification is not changed and consists in comparing $\hat{e}(R, S)$ to $\hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{ID})^{H_3(R)}$. The two values must be equal in order to consider the signature as valid.

• When Hess signature scheme is chosen, the user picks an arbitrary point $P_1 \in G_1^*$ and a random $k \in \mathbb{Z}_q^*$. In addition, it executes the following steps:

1) $r = \hat{e}(P_1, P)^k$
2) $v = H_4(M, r)$
3) $U = v \cdot Priv_{ID} + r_2^{-1} \cdot k \cdot P_1$

The signature is formed by the pair $(U, v) \in G_1 \times \mathbb{Z}_q^*$.

The signature verifier then has to compute:

1) $r = \hat{e}(U, r_2 \cdot P) \cdot \hat{e}(Pub_{ID}, -P_{pub})^v$
2) The signature is accepted if and only if $v = H_4(M, r)$

Any ID-based signature can be changed to take into consideration the presence of the *token*. In fact, these modification can be applied to any signature scheme that relies on a pairing function $\hat{e}$. We only have to add the secret r_2 to the signature generation process. In addition, we have to modify the signature verification algorithm in order to use $(r_2 \cdot P)$. The point is that r_2 and r_2^{-1} disappear when multiplication is used in G_2.

C. Security Discussion

In this section, we present how the aforementioned authentication protocol removes some attacks and how it can be enhanced to remove other threats:

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

The use of T_0 at the MA level removes DoS attacks but does not help against Distributed DoS attack (DDoS). In fact, an attacker may control many STAs and launch a DDoS attack against the AS by sending many *Start-authentication* messages corresponding to different (zom-

bie) STAs under control. The aim of the attack is to flood the AS with a big amount of authentication requests. That is why, we proposed to use the second threshold T_1 at the level of the AS.

• *Replay attack:* The random number initially selected combined with the timestamp value serve to prevent from replay attacks. In fact, after receiving the Beacon from its peer MA, the supplicant STA receives a timestamp Delta value Δ. This Δ is used to verify the freshness of the received and sent message as presented in [13]. For example, when a supplicant STA sends its first *Start-authentication* message containing a nonce n_1 and a timestamp t_1, the receiving AS compares the reception time (t_{recep}) of the message to the timestamp t_1 using Δ as follows: $|t_{recep} - t_1| < \Delta$. If this inequality does not hold, the AS rejects the received message. In addition, every STA stores the last reception time and timestamp values received from its peers for future timestamp verification. Even with the use of Δ, a window of vulnerability for replay attacks exists until the timestamp expires. That is why we introduced the nonce n_1 in the first message. In fact, as n_1 is randomly selected, it adds more freshness to the message. In addition, we impose the rule that n_1 must be renewed for every authentication request sent during a Δ period which starts when the first authentication request is sent. We assume that this random number is at least 128 bit length which implies that the probability of getting the same random number for two consecutive authentications is equal to $1/2^{128}$. For the upcoming message, n_1 will be used to initiate a sequence number which increases the prevention against replay attacks.

• *Private key recovery by an attacker*: Concerning the private key recovery, an attacker needs first to recover the partial private key of the STA. He may get the encoded private key of a supplicant but he has to find the secret r_1 in order to recover the partial private key of the supplicant. The problem of finding r_1 is equivalent to the discrete logarithm problem over an elliptic curve group (ECDLP) [2]. In addition, the full private recovery requires from an attacker to guess r_2 from $(r_2 \cdot P)$ value which comes also to solve the ECDLP.

• *Key escrow attack*: In order to avoid the key escrow problem, every STA participates to the private key generation with a secret value r_2. This secret is included in the public *token* but after being multiplied by the point P. So neither the AS nor the MKD are able to compute the private key corresponding to $(r_2 \cdot P)$. Note that the AS can technically generate a fake *token* with a fake $(r'_2 \cdot P)$ in order to impersonate as STA, however, this is in contradiction with the assumption that the AS must be trustworthy.

D. Protocol enhancement

The proposed authentication mechanism can be enhanced to include the usage of a secret value s_i corresponding to each MKD_i. This implies that every MKD domain will be differentiated by its own *public elements*. The distribution of the secrets s_i to the MKD_i is controlled by the AS. This modification may be interesting in the case where the AS wants to localize every STA using its key pair or to identify STA movements during handover process.

In addition, the public key of STA may become dynamic by modifying the public key generation as follows: $Pub_{ID} = H(ID \| r_2 \cdot P)$. In fact, the public key becomes bound to its corresponding private key because it will contain $(r_2 \cdot P)$.

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 removing 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). Our future works are focusing on validating the proposed protocol and testing its performance in terms of time and computation power. Also evaluation of its scalability and ease of deployment is part of our perspectives.

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

[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] 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.

[6] 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.

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

[8] 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.

[9] L. Wenju, S. Yuzhen, and W. Ze, "A wireless mesh network authentication method based on identity based signature," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, 24-26 2009, pp. 1 –4.

[10] *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.

[11] 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

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

[13] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)," RFC 3971 (Proposed Standard), Mar. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc3971.txt