On the use of ID-based cryptography for the definition of new EAP authentication methods

Aymen Boudguiga, Maryline Laurent

On the use of ID-Based Cryptography for the definition of new EAP authentication methods

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*— We propose in this paper, two ID-Based authentication methods for the Extensible Authentication Protocol (EAP), as an alternative to methods relying on Public Key Infrastructure (PKI), to provide nodes with private and public keys. ID-Based Cryptography (IBC) proposes to derive the public key from the node's identity directly. As such, there is no need for deployment of a Certification Authority (CA) and the burdensome management of certificates is removed. IBC relies on a Private Key Generator (PKG) for the computation of stations private keys.**

Our first presented authentication method corresponds to a situation where the PKG is trustful. As such, the PKG generates the private keys of all the network stations. However, our second contribution presents an authentication method which is resistant to the Key Escrow Attack. That is, we make each station generate its own ID-Based private key. In addition, results from implementation tests are given and prove how efficient IBC might be for use in wireless networks.

I. Introduction

Nowadays, most of the authentication schemes proposed for wireless networks rely on the 802.1X standard [1]. The 802.1X standard was designed to integrate Extensible Authentication Protocol (EAP) into IEEE 802 wired networks. It gave also birth to the IEEE 802.11i [2] which is more specific to wireless networks but still refers to similar authentication methods. The 802.11i standard is the main standard used to secure 802.11 WLAN and IEEE wireless mesh networks specified in the 802.11s standard [3].

The authentication methods proposed by the 802.1X standard are based either on the verification of a secret shared between two stations or a signature mechanism that lies on certificates to prove the ownership of a public key and a signature for proving knowledge of the associated private key. The management of public and private keys requires deploying CAs to control the generation, revocation and duration of certificates. This system is disadvantageous in wireless environments, such as adhoc and wireless sensor networks, where stations may have some power and memory constraints and where CA reachability is not guaranteed.

In this paper, we present two EAP authentication methods adapted to wireless networks and using ID-Based Cryptography (IBC). IBC considers the station identity as its public key, and makes it possible to derive a corresponding private key. This derivation function, as well as the secure transmission of the private key to its owner are performed by the Private Key Generator (PKG). We propose in this work to use the network Authentication Server (AS) as a PKG. As such, each station (STA) has to authenticate itself to the AS before requesting the computation of its private key.

Note that IBC requires lightweight implementations at the client level. Compared to Public Key Infrastructure (PKI) certificate management, it does not need any special space for certificate storage, and the key revocation operation is simpler. Key revocation in IBC is bound to a validity period which is defined either by the PKG or the connecting station. Please refer to the work [4] for a good comparison between PKI and IBC.

The major problem of IBC is the Key Escrow Attack (KEA) which can be attempted by the PKG. The KEA attack refers to the attack where a PKG impersonates as a legitimate station due to its (i.e., the PKG's) knowledge of the station private key. That is, the PKG is able to decipher any encrypted message transmitted to (or by) a station. In addition, the PKG has the capability to sign a message on behalf of any station due to its knowledge of this station private key.

Our first contribution is an authentication method which is adapted to a network where the PKG is supposed to be trustful. That is, the PKG is not going to attempt a KEA. This authentication serves to mutually authenticate a station with the AS. We called this first authentication method the EAP ID-Based Authentication method (EAP–IBA).

Our second contribution presents an authentication method which permits to avoid the KEA. The idea is that each station is going to generate its own private key. Meanwhile, the PKG generates a *token* to prove the STA ownership of the private key. We will be referring to this authentication method by the EAP Key Escrow Resistant ID-Based Authentication method (EAP–KERIBA).

The remainder of this work is organized as follows. First, the ID-based encryption and signature mechanisms are introduced in Section II. Then, we describe the EAP–IBA method in Section III-A and the EAP–KERIBA method in Section III-B. We next discuss the security properties of these two authentication methods in Section III-C. Finally, the implementation of some ID-based signature and encryption schemes gives performance results in terms of

computation time and memory capacity related to pairing functions (Section IV). By the way, the ID-based signature schemes are presented comparatively to the famous RSA and ECDSA performances.

II. ID-Based Cryptography

ID-Based Cryptography (IBC) was initially introduced by Shamir [5] to provide entities with public and private key pairs with no need for certificates, Certification Authority (CA) and PKI. Shamir assumes that each entity uses one 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, IBC has been enhanced by the use of the Elliptic Curve Cryptography (ECC) [6]. As a consequence, new ID-Based encryption and 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 IBC requires lightweight implementations at clients. Compared to PKI certificate management, there is no need for storing certificates, and the key revocation operation is much simpler. Key revocation in IBC is bound to a validity period which is defined by the PKG or chosen by the station and acknowledged by the PKG.

Sometimes, certificates are considered as IBC as they bind the user's public key to his identity. In this paper, note that IBC is considered as the cryptographic schemes where the public key is computationally derived from the identity. That is, the public key is the output of a function (mostly a hash function) that takes as input the user's identity. There are many existing types of IBC schemes but this article has only interest in the ones using pairing functions.

In the following sections, we present the key generation processing for IBC. Furthermore, we introduce some well known ID-Based Encryption and Signature (IBE and IBS) schemes which have been verified secure with the random oracle model [7].

A. ID-Based key generation

When a station needs a private key, it provides the 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 [8]. For generating these parameters, the PKG runs a Probabilistic Polynomial Time 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 $\mathbb{G}_T$. Paterson et al. [9] defined three types of pairing functions that can be divided into two families:

1) Symmetric pairing: it verifies $\mathbb{G}_1 = \mathbb{G}_2$.
2) Asymmetric pairing: it verifies $\mathbb{G}_1 \neq \mathbb{G}_2$. This pairing function can be further classified based on the existence (or not) of an efficient homomorphism $\phi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$.

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 (or any efficient pairing) [10].

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 the PKG to the network users because they are needed during the public key derivation and the cryptographic operations.

The key derivation operation starts when the PKG receives the ID of the node that is requesting a private key (Figure 1). 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: $Priv_{ID} = f(s, Pub_{ID})$. 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 $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, g, P, P_{pub}, H_1, \ldots, H_k\}$.

It is clear from the aforementioned key derivation scheme that the 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. To mitigate the key escrow attack (KEA), a strong assumption is made necessary that the PKG is a trustworthy entity. In the following, we present some IBE and IBS schemes.

B. Boneh and Franklin encryption scheme

Boneh and Franklin (BF) proposed in 2001 [10] an IBE scheme using ECC and a symmetric pairing function. They define two hash functions H_1 and H_2 such that: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$. So BF public

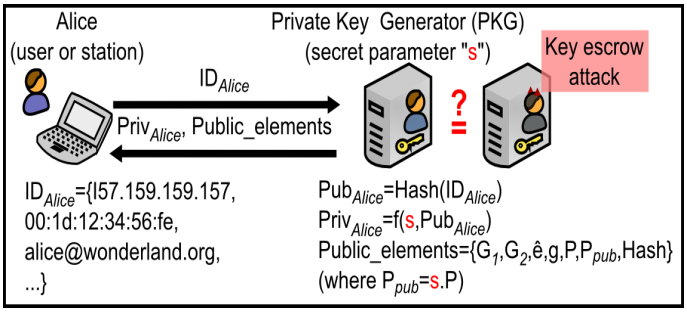

Fig. 1: ID-Based key generation.

elements are $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, g, P, P_{pub}, H_1, H_2\}$. 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 $s \in \mathbb{Z}_q^*$.

To encrypt a message $M \in \{0,1\}^n$ using the public key Pub_{ID}, a user generates a secret random $k \in \mathbb{Z}_q^*$ and computes the ciphertext C as $C = (U, V) = (k \cdot P, M \oplus H_2(\hat{e}(Pub_{ID}, P_{pub})^k))$. The decrypting entity deciphers the received message as follows: $M = V \oplus H_2(\hat{e}(Priv_{ID}, U))$.

C. Paterson signature scheme

Paterson proposed in 2002 [11] an IBS scheme using ECC and a symmetric pairing function. He defines three hash functions H_1, H_2 and H_3 such that: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_3 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$. So, Paterson public elements are $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, g, P, P_{pub}, H_1, H_2, H_3\}$. 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 $s \in \mathbb{Z}_q^*$.

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 $(R, S) \in \mathbb{G}_1 \times \mathbb{G}_1$ 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.

III. EAP ID-Based Authentication methods

The Extensible Authentication Protocol (EAP) [12] was originally defined as an extension to the Point to Point Protocol in order to provide a mechanism for the selection of authentication methods. Then it has evolved to become the standard that is used for station authentication in the existing networks. Particularly, it has been adapted to the 802.11 networks architectures in the IEEE 802.11i standard [2] which extends the IEEE 802.1X specification [1]. The IEEE 802.1X defines the EAP over LAN protocol (EAPOL).

When a station (STA) joins the network for the first time or after being disconnected for a while, it authenticates itself to one of its 1 hop neighbours that acts as an authenticator. That is, through the exchange of some request and response messages, the authenticator and the supplicant negotiate the authentication method which is going to be used. When the authenticator does not support the authentication method proposed by the supplicant STA, it acts as a passthrough server to transmit STA authentication messages to a backend Authentication Server (AS). Generally, the AS implements the most known authentication methods. At the end of an authentication, the AS transmits to STA the result of the authentication using authentication success and authentication failure messages.

We propose in this section two different EAP authentication methods that rely on IBC but correspond to two different situations in practice. We assume first to use the AS as a PKG in the two methods. We do so to avoid the deployment in our network of a new entity in order to use it as a PKG. As such, the AS will be the responsible of STA authentication and key derivation. In addition, we assume that the AS and STA share a secret password *pwd*. This *pwd* permits to authenticate them mutually.

Our first contribution (Section III-A) corresponds to a situation where STA entrusts the AS for its private key generation. That is, STA uses the EAP–IBA method to authenticate to the AS. Then, the AS computes the private key of STA ($Priv_{STA}$). STA assumes that the AS will not try a KEA even if it knows its $Priv_{STA}$. However, we consider in our second contribution (Section III-A) the opposite situation where STA does not trust the AS for its private key computation. We assume that the AS generates only the domain public elements. Then, STA executes the EAP–KERIBA method to authenticate to the *AS*. During the authentication, STA receives the IBC public elements from the AS and uses them to compute its own private key locally. As such, the AS will not know the private key of STA and so it will not be able to realize a KEA.

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. For example, if we consider that we are using BF encryption scheme and Paterson signature algorithm during the protocol execution, the public elements that AS has to generate are: $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, g, P, P_{pub}, H_1, H_2, H_3, H_4\}$ where: $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ and $H_4 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$.

A. EAP ID-Based Authentication method (EAP–IBA)

For straight integration of IBC into wireless networks, we defined the EAP ID-Based Authentication method (EAP–IBA). In accordance with the EAP protocol [12], four types of message are used: request, response, success and failure. The request message is always sent first from the AS (or the authenticator) to the supplicant STA. It contains information about the authentication method to be used. The next request messages support the authentication exchanges. The response messages are generated by the supplicant STA in response to the requests. They contain the supplicant authentication elements. Finally, the success or failure messages are sent from AS to STA at the end of the authentication.

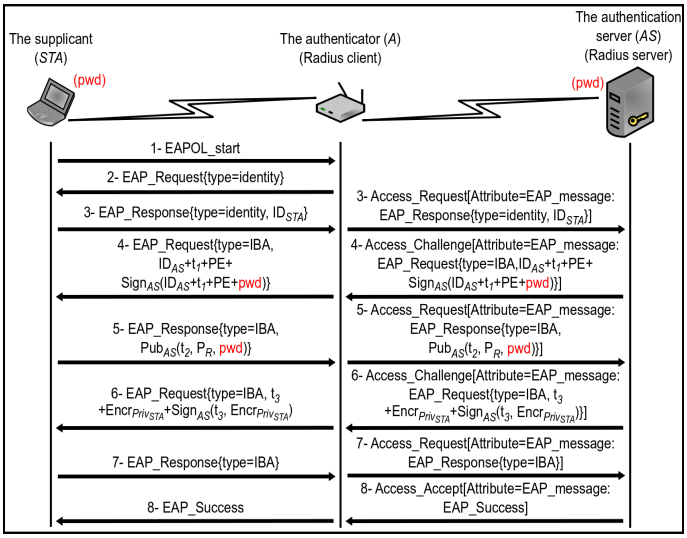

Fig. 2: EAP ID-Based Authentication Method (EAP–IBA)

The EAP–IBA method is illustrated in Figure 2. EAP messages are encapsulated between STA andthe authenticator into EAPOL (EAP Over LAN) packets using IEEE 802.1X; they are encapsulated between the authenticator and the AS into AAA (Authentication Authorization, Accounting) messages like RADIUS or Diameter for Wi-Fi networks, or into the Mesh EAP Message Transport Protocol for mesh networks [3].

When a STA joins the network for the first time of after being disconnected for a while, it starts an authentication with the AS. STA must first send an *EAPOL start* message to one of its 1-hop authenticators which responds with an EAP request of type *identity* (Message 2 in Figure 2). Upon receiving the request, STA sends an EAP response message to the authenticator. The response is of type *identity* and contains STA identity ID_{STA} (Message 3). The authenticator transmits this response to the AS.

When the AS receives the ID_{STA}, it searches in its password database for the password *pwd* corresponding to ID_{STA}. Then, the AS generates the Message 4. The latter is an EAP request message of type EAP–IBA. It contains the identity of the AS (ID_{AS}), a timestamp t_1, the AS public elements (PE) and a signature of the concatenation of these fields and the *pwd*. For example, if the AS uses Paterson signature scheme, its signature is formed by the pair $(R, S) \in \mathbb{G}_1 \times \mathbb{G}_1$ such that:

$$R = k.P, \quad k \in \mathbb{Z}_q^*$$
$$S = k^{-1}(H_3(t_1||ID_{AS}||PE||pwd).P + H_4(R).Priv_{AS})$$

Upon receiving the Message 4, STA gets from the public elements (PE) the hash function in use for the public key computation (H_1 in our example). Then, STA computes the public key of AS as $Pub_{AS} = H_1(ID_{AS})$. In addition, it concatenates the password *pwd* to ID_{AS}, t_1 and the public elements which have been received in clear. That is, STA creates the message that has been signed by the AS. At this point, STA uses the computed Pub_{AS} to verify the received signature. In our example, the signature verification consists in verifying that:

$$\hat{e}(R, S) = \hat{e}(P, P)^{(t_1||ID_{AS}||PE||pwd)}.\hat{e}(P_{pub}, Pub_{AS})^{H_4(R)}$$

If the signature verification fails, STA deduces that the received public elements were wrong or have been modified during the transfer of the Message 4. The reception of wrong public elements implies that Pub_{AS} which STA computed is not valid. The verification failure can also result from the usage of an invalid *pwd*. Thus, the supplicant STA does not authenticate the AS unless a valid signature is received in the Message 4. The authentication process has to be stopped when the signature verification fails.

If the signature verification is successful, STA authenticates the AS and the public elements. Consequently, STA creates the Message 5 to authenticate itself to the AS using its *pwd*. The Message 5 is an EAP response message of type EAP–IBA. It contains the encryption with Pub_{AS} of a timestamp t_2, a point $P_R = r_{STA}.P$ and the password *pwd*. The AS needs the point P_R for later encryption of STA private key. P_R is computed using a secret value r_{STA} which is randomly selected in $\mathbb{Z}_q^*$. For example, if BF encryption algorithm is selected, the encrypted message will be:

$$C = (U, V) = (k.P, (t_2||P_R||pwd) \oplus H_2(\hat{e}(Pub_{AS}, P_{pub})^k))$$

Upon receiving the Message 5, the AS deciphers it with its private key $Priv_{AS}$. In our case, the AS makes the following operation:

$$M = t_2||P_R||pwd = V \oplus H_2(\hat{e}(Priv_{AS}, U))$$

Therefore, the AS authenticates STA by verifying the validity of the *pwd*.

If STA authentication succeeds, the AS computes the STA private key as $Priv_{STA} = s.Pub_{STA}$ where s is the AS secret value and $Pub_{STA} = H_1(ID_{STA})$ is the public key of STA. Then, the AS ciphers $Priv_{STA}$ as $Encr_{Priv_{STA}} = Priv_{STA} + s.P_R$ and creates the Message 6. The latter contains a timestamp t_3, the encrypted private key of STA ($Encr_{Priv_{STA}}$) and a signature over the concatenation of these fields. Thus, the AS signature is $(R, S) \in \mathbb{G}_1 \times \mathbb{G}_1$ where:

$$R = k.P, \quad k \in \mathbb{Z}_q^*$$
$$S = k^{-1}(H_3(t_3||Encr_{Priv_{STA}}).P + H_4(R).Priv_{AS})$$

At the reception of the Message 6, STA uses the AS public key to verify the signature. If the signature verification is successful, STA recovers its private key $Priv_{STA}$ from the $Encr_{Priv_{STA}}$ by computing:

$$Priv_{STA} = Encr_{Priv_{STA}} - r_{STA}.P_{pub}$$

Note that $-r_{STA}.P_{pub}$ is the inverse of $r_{STA}.P_{pub}$. Then STA computes its public key $Pub_{STA} = H_1(ID_{STA})$ and verifies the equality:

$$\hat{e}(Pub_{STA}, P_{pub}) = \hat{e}(Priv_{STA}, P) \qquad (1)$$

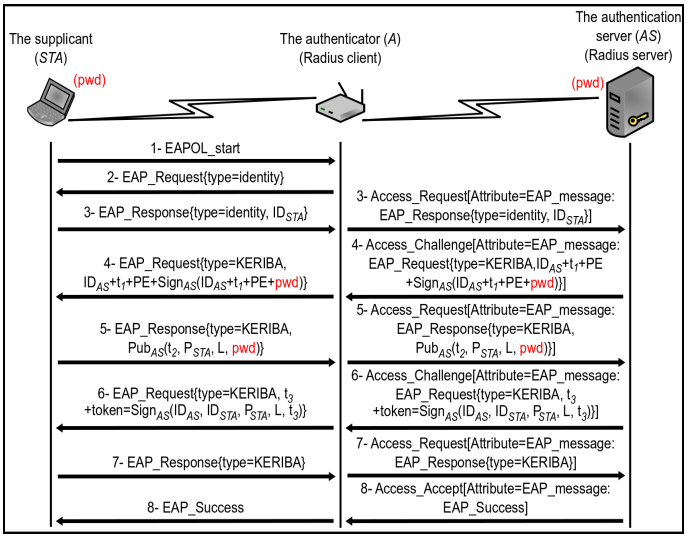

Fig. 3: EAP Key Escrow Resistant ID-Based Authentication method (EAP–KERIBA).

If this verification holds, STA authenticates the AS as the generator of $Priv_{STA}$ because the Equation 1 binds $Priv_{STA}$ to Pub_{STA} using the points P and P_{pub} which are parts of the public elements received in the Message 4. Recall that these public elements have been already authenticated at the reception of the Message 4.

STA responds to the AS with an empty EAP response of type EAP–IBA to acknowledge the good reception of its private key $Priv_{STA}$. Finally, the AS sends to STA an EAP success packet to indicate the success of their mutual authentication.

B. EAP Key Escrow Resistant ID-Based Authentication method (EAP–KERIBA)

In this section, we assume that the AS is acting as PKG but only for the generation of the public elements that STAs use to derive their private and public keys. In order to decrease the risk of a KEA, we make every STA generate its own private key locally while AS computes a *token* which contains pieces of information that are bound to STA private key. In addition, we assume as for EAP–IBA that the AS and STA share a secret value (*pwd*) that they use with IBC to mutually authenticate.

EAP–KERIBA is used by a STA when it joins the network for the first time or after being disconnected for a while. To perform an authentication, STA must first get the public elements that are published by AS (acting in this proposal as a PKG which only computes these public elements). After a successful initial authentication, STA computes its private key corresponding to its ID-Based public key. For later authentications to other STAs, STA can use its *token* and any existing authentication scheme based on asymmetric cryptography.

When STA wants to authenticate to the AS, it sends an *EAPOL Start* to its authenticator (Message 1 in Figure 3). Upon receiving this message, the authenticator responds with an EAP request of type *identity* to recover the STA identity (Message 2). Consequently, STA responds with an EAP response of the same type containing its identity ID_{STA} (Message 3). The authenticator transfers then the message 3 to the AS which starts an EAP authentication of type EAP–KERIBA with STA.

When the AS receives ID_{STA} in Message 3, it searches in its password database for the password corresponding to this STA. Then it generates the Message 4. The password is included in this message in order to authenticate the AS with its public elements. The Message 4 is an EAP request of type EAP–KERIBA. It contains the identity of the AS (ID_{AS}), a timestamp t_1, the public elements (PE) and a signature of the previous fields after concatenating them to the password (pwd). For example, if the AS uses the Paterson signature scheme, the AS signature is the pair $(R, S) \in \mathbb{G}_1 \times \mathbb{G}_1$ such that $R = k \cdot P$, $S = k^{-1}(H_3(ID_{AS}||t_1||PE||pwd) \cdot P + H_4(R) \cdot Priv_{ID})$ where k is a random integer.

Upon receiving the Message 4, STA gets from the public elements the hash function in use for the public key computation (H_1 in our example). It computes the public key of AS as $Pub_{AS} = H_1(ID_{AS})$. It concatenates the password pwd to ID_{AS}, t_1 and the public elements to form the message that has been signed. At this point, STA uses the computed Pub_{AS} to verify the received signature. In our example, the signature verification consists in comparing $\hat{e}(R, S)$ to $(\hat{e}(P, P)^{H_3(ID_{AS}||t_1||PE||pwd)} \cdot \hat{e}(P_{pub}, Pub_{AS})^{H_4(R)})$. If the signature verification is successful, STA authenticates the AS and the public elements. Otherwise, STA stops the authentication.

If the signature is valid, STA has to authenticate itself to AS using the preshared secret and creating the Message 5 (the EAP response which corresponds to the Message 4). The latter contains the encryption with Pub_{AS} of a timestamp t_2, a point $P_{STA} = s_{STA} \cdot P$, a lifetime L and the password pwd. The point P_{STA} is used to avoid the KEA. Its usage is detailed in the upcoming section III-C. It is computed using a secret value s_{STA} which is randomly selected in $\mathbb{Z}_q^*$. This secret s_{STA} enables STA to compute its own private key $Priv_{STA}$ such that $Priv_{STA} = (s_{STA} \cdot Pub_{STA})$. As such, STA does not rely any longer on the PKG (namely AS) to compute its private key. The lifetime L refers to the validity duration that STA wishes to acquire for its P_{STA}. It indicates also the lifetime that STA chooses for its own private key $Priv_{STA}$ because $Priv_{STA}$ is bound to P_{STA} through the secret value s_{STA}. For example, if BF encryption algorithm is selected, the encrypted message is $C = (U, V) = (k \cdot P, (t_2||P_{STA}||L||pwd) \oplus H_2(\hat{e}(Pub_{AS}, P_{pub})^k))$ where k is a random number.

Upon receiving this fifth message, the AS uses its private key $Priv_{AS}$ for deciphering. In our case, the AS makes the following operation $M = t_2||P_{STA}||L||pwd = V \oplus H_2(\hat{e}(Priv_{AS}, U))$. In addition, the AS authenticates STA by verifying the value of password pwd. If the AS succeeds in authenticating STA, it generates the Message 6. This message contains a timestamp t_3 and a *token*. The *token* represents the signature of the AS over the following fields:

the identity of AS (ID_{AS}), the identity of STA (ID_{STA}), the lifetime L and the timestamp t_3. The validity of the private key $Priv_{STA}$ starts at t_3 for L duration.

At the reception of Message 6, STA verifies the AS signature over the *token*. If the verification is valid, STA responds to AS with an empty EAP response of type EAP–KERIBA to acknowledge the good reception of the Message 6. Finally, the AS sends to STA an EAP success packet to indicate the success of their mutual authentication.

C. Security discussion

In this section, we present informally how the aforementioned EAP authentication protocols remove some attacks. We use an 'active saboteur' attacker model as defined by Dolev and Yao [13]. That is, an attacker might be a user of the network and can have access to all the traffic. We suppose also that the ID-Based signature and encryption algorithms used for our EAP–IBA and EAP–KERIBA schemes have been already verified secure in the random oracle. That is, we will be only interested in discussing the security properties of the exchanged messages.

• *Replay attack:* For avoiding replay attacks, we make use of timestamps t_1, t_2 and t_3. However, timestamp usage assumes that all the stations in the network are synchronized. This can be done with the Beacon frames received by a STA from its 1-hop neighboring APs in 802.11 or 1-hop peer STAs in 802.11s. These Beacons include time information about their senders' clock values. In addition, they contain a certain value Δ which is a time interval used for the verification of message freshness.

The timestamp Delta value Δ is chosen by the network administrator and is used as follows: During the execution of EAP–IBA or EAP–KERIBA, the first timestamp t_1 is included by the AS in the Message 4, the receiving supplicant STA 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, STA rejects the received message. In addition, every STA stores the last reception time and timestamp values received from its peers for future timestamp verification.

• *Collision attack:* An attacker can try making a collision over the second message in order to get the password or to impersonate as the AS. If the collision attack is successful, he gains access to STA secret information and original password. Thanks to the known birthday attack, it is known that $2^{n/2}$ attempts are necessary for getting a collision over a n-bit length hash.

This attack may be avoided by changing the password periodically to decrease the risk of collision. However, the best solution in practice is to use a one-time password. That is, the AS and STA share a master session key which is used to derive a new password for each authentication session.

• *Private key recovery from STA encrypted private key:* During the execution of EAP–IBA, an attacker can get the enciphered private key $Encr_{Priv_{STA}}$ of a supplicant STA by realizing just a capture of traffic. However, it has to find the secret r_{STA} in order to recover $Priv_{STA}$. To do so, the attacker has to know $Priv_{AS}$ in order to decipher the Message 5. Then, he has to guess r_{STA} from $P_R = r_{STA}.P$. Therefore, he has to solve the Elliptic Curve Discrete Logarithm Problem (ECDLP) in the group $\mathbb{G}_1$ of elliptic curve points [6]. Or, the ECDLP becomes a hard cryptographic problem when the order q of $\mathbb{G}_1$ is at least 160 bit long. That is, the complexity of solving the ECDLP is around $\mathcal{O}(2^{80})$ [6].

When we look in depth at the encryption of $Priv_{STA}$ into $Encr_{Priv_{STA}}$, we notice that it is equivalent to ElGamal encryption algorithm when it is applied in an additive group [14]. In that case, r_{STA} is the private key of STA and $r_{STA}.P_{pub}$ is its corresponding public key.

Upon deciphering the Message 5 of EAP–IBA, the AS shares $P_R = r_{STA}.P$ with STA. So that, the AS becomes the unique entity which is able to compute the point $r_{STA}.P_{pub}$ used for $Priv_{STA}$ encryption. The AS computes $r_{STA}.P_{pub}$ by multiplying P_R with its secret s. Then, the AS sums $r_{STA}.P_{pub}$ to $Priv_{STA}$ to get $Encr_{Priv_{STA}}$. In practice, the AS computes $Priv_{STA}$ and encrypts it as $Encr_{Priv_{STA}}$ only by doing one scalar/point multiplication. That is, the AS adds $r_{STA}.P$ to $Pub_{STA} = hash(ID_{STA})$. Then, it multiplies the result of the sum by its secret s to get:

$$\begin{aligned} Encr_{Priv_{STA}} &= s.(r_{STA}.P + Pub_{STA}) \\ &= r_{STA}.s.P + s.PubSTA \\ &= r_{STA}.P_{pub} + Priv_{STA} \end{aligned}$$

At the reception of Message 6 of EAP–IBA, STA recovers its $Priv_{STA}$ from $Encr_{Priv_{STA}}$ by calculating:

$$\begin{aligned} Priv_{STA} &= Encr_{Priv_{STA}} - r_{STA}.P_{pub} \\ &= r_{STA}.P_{pub} + Priv_{STA} - r_{STA}.P_{pub} \end{aligned}$$

STA is the unique entity which is able of deciphering $Encr_{Priv_{STA}}$ because it is the unique station which is in possession of r_{STA}.

• *Key escrow attack:* During the EAP–KERIBA execution, we supposed that each STA is generating its own private key $Priv_{STA}$ based on the use of a secret s_{STA}. This secret is also used to compute a point $P_{STA} = s_{STA} \cdot P$. The point P_{STA} is included into the *token* generated by the AS on behalf of STA. Consequently, the AS (acting as the PKG) is not able to make a KEA because it has not generated STA's private key ($Priv_{STA}$). In addition, the only way to recover STA's private key would be to compute s_{STA} from $P_{STA} = s_{STA} \cdot P$ which is equivalent to solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) [6]. The risk of a KEA is reduced to the unique case where STA is offline and AS generates a fake *token* on behalf of STA for impersonation of it. Note that this issue is common to any ID-Based cryptograms as the PKG can always impersonate as an offline legitimate STA.

During EAP–KERIBA execution, we made each STA generate its own $Priv_{STA}$ contrarily to EAP–IBA. So, STA does not rely any longer on the AS to derive its

private key. However, it has to prove the ownership of $Priv_{STA}$ to the AS and to other STAs. Consequently, we decided to introduce the point P_{STA} to justify a key ownership by a STA. P_{STA} is computed by STA using the same secret s_{STA} that has been used for $Priv_{STA}$ computation. The point P_{STA} is then authenticated by the AS when it is received with the *pwd* in the Message 5 of EAP–KERIBA. It is included in the *token* of STA and signed by the AS. This point P_{STA} has to replace P_{pub} when ciphering a message addressed to STA or when verifying a signature from STA. In fact, we are not going to use P_{pub} for all the STA but only when verifying the AS signatures or when ciphering a message for the AS. For the other STAs, we use the corresponding point P_{STA}. For example, when STA1 receives a signature from STA2, it has to use P_{STA2} and Pub_{STA2} to verify the signature. STA1 authenticates P_{STA2} because it is included in the $token_{STA2}$ which is signed by the AS. In addition, STA1 gets Pub_{STA2} by hashing ID_{STA2}.

Through the encryption or the signature of a message, STA has to prove the association between its $Priv_{STA}$ and the computed P_{STA} which is signed by the AS. This association implies directly that STA is the owner of the private key $Priv_{STA}$ and that it has been initially authenticated by the AS. For example, if we are using the aforementioned Paterson signature scheme, the public elements generated by the AS are: $\{\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, g, P, P_{pub} = s \cdot P, H_1, H_2, H_3\}$ and $Priv_{AS} = s \cdot Pub_{AS}$ where $Pub_{AS} = H_1(ID_{AS})$. The signature of a message M by the AS is the pair $(R, S) \in \mathbb{G}_1 \times \mathbb{G}_1$ where: $R = k \cdot P$, $S = k^{-1}(H_2(M) \cdot P + H_3(R) \cdot Priv_{AS})$ and k is a random scalar. The verification of this signature consists in comparing $\hat{e}(R, S)$ to $(\hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{AS})^{H_3(R)})$. The verification holds because:

$\hat{e}(R, S) = \hat{e}(k \cdot P, k^{-1}(H_2(M) \cdot P + H_3(R) \cdot Priv_{AS}))$
$\Longrightarrow \hat{e}(R, S) = \hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(P, Priv_{AS})^{H_3(R)}$
$\Longrightarrow \hat{e}(R, S) = \hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(P, s \cdot Pub_{AS})^{H_3(R)}$
$\Longrightarrow \hat{e}(R, S) = \hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(s \cdot P, Pub_{AS})^{H_3(R)}$
$\Longrightarrow \hat{e}(R, S) = \hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(P_{pub}, Pub_{AS})^{H_3(R)}$

If STA wants to make the same signature, it uses the point P_{STA} and its private key ($Priv_{STA} = s_{STA} \cdot Pub_{STA}$) instead of the P_{pub} and $Priv_{AS}$ when signing a message using the Paterson signature algorithm. The signature verifier has to get P_{STA} from the *token*, and then it has to compare $\hat{e}(R, S)$ to $(\hat{e}(P, P)^{H_2(M)} \cdot \hat{e}(P_{STA}, Pub_{STA})^{H_3(R)})$.

• *Denial of service attack (DoS):* To avoid an attacker making a DoS attack against the AS by sending a big amount of *Start authentication* messages, we decided to limit at the AS the number of authentication requests to a threshold T. That is, the authenticator accepts only one authentication request per supplicant STA. Then STA has to wait for the AS response. In addition, the AS limits the number of authentication requests coming from the same authenticator to T. When the number of authentication requests exceeds T, the AS drops all the upcoming packets received from that authenticator.

TABLE I: IBS and IBE elementary operations.

IBS or IBE scheme	$\mathbb{G}_T$ Exp	Pt/scalar mul	Pairings
Paterson signature	0	4	0
Paterson verification	2	0	3
Hess signature	1	2	1
Hess verification	1	0	2
Barreto et al. signature	1	1	0
Barreto et al. 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

IV. Implementations and results

In this section, we present the implementation results related to our authentication schemes. We focus on the time performance of our EAP–IBA and EAP–KERIBA methods. That is, we have noticed that our EAP methods performance depends mostly on the time consumption of the used signature and encryption algorithms in messages 4 and 5. Consequently, we decided to evaluate the time performance of a set of IBS and IBE in order to elect the most suited IBS and IBE for EAP–IBA and EAP–KERIBA.

First, we show how to evaluate the security level of an ID-Based signature or encryption scheme. That is, the studied IBS and IBE performances are compared at the same security level. In the sequel, we present results of the comparison of the time performances of different ID-Based signature and encryption algorithms. The studied IBS are compared by the way to RSA and ECDSA. Finally, we study the gain in memory that STAs realize when IBC replaces the use of certificates. We show that IBC is more efficient in terms of memory consumption than classical PKI where certificates are used and need to be stored.

A. Security level equivalence between schemes

To compare the performances of IBS or IBE schemes, a first analysis of the number of mathematical operations can be done. Barreto et al. [15] used $\mathbb{G}_T$ exponentiations, scalar point multiplications (Pt/scalar mul) and pairing computation operations to evaluate the signature scheme performances. Table I presents a comparison of Paterson signature, Barreto et al. signature and Hess signature [16] based on elementary operations as in [15]. In addition, it presents a comparison of the following encryption schemes: BF, Boneh and Boyen (BB) [17] and Chen et al [18]. For a fair comparison between an IBS and an existing signature scheme like RSA, we need defining the security level of each scheme and making the comparison for the same security level.

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

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

2^k [9]. In asymmetric cryptography, the security level of an algorithm is defined with respect to the hardness of factoring prime numbers (the case of RSA) or solving the Discrete Logarithm Problem (DLP) in an additive group (the case of ECDSA). This concept of security level sets the length in bits of RSA keys and EC keys. 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.

Let $E(\mathbb{F}_p)$ denote the elliptic curve defined over the finite prime field $\mathbb{F}_p$. $\mathbb{G}_1$ and $\mathbb{G}_2$ correspond mostly to the q-torsion subgroups of $E(\mathbb{F}_p)$ and $E(\mathbb{F}_{p^k})$ where k is the embedding degree of the curve $E(\mathbb{F}_p)$ relatively to q. Meanwhile, $\mathbb{G}_T$ is a multiplicative subgroup of $\mathbb{F}_{p^k}$ of order q [6]. For example, assume that the prime order p of $\mathbb{F}_p$ is 512 bits long, the order q is 160 bits length while the embedding degree relatively to q of the curve $E(\mathbb{F}_p)$ is 2. The pairing function $\hat{e}$ is then defined over the subgroups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of order q. The security level of $\hat{e}$ is defined respectively to the hardness of solving DLP in $\mathbb{G}_T$. As $\mathbb{G}_T$ is a subgroup of $\mathbb{F}_{p^2}$ which has an order of 1024 bits, DLP hardness in $\mathbb{G}_T$ is defined respectively to this 1024 bits order. That is, $\hat{e}$ security level is equivalent to an RSA key of 1024 bits length, and so, with respect to Table II, to a security level of 80 bits.

B. Time performances

As the time performance of EAP–IBA and EAP–KERIBA is mostly depending on the time consumption of the signature and encryption algorithms in use, a comparison is established between the time consumption of some IBS and IBE algorithms. The three signature schemes selected are the following: Paterson signature, Barreto et al. signature [15] and Hess signature [16]. For illustration, we also include time consumption of the RSA and ECDSA signature algorithms. In addition, we compared BF encryption to BB encryption [17] and Chen et al. encryption algorithm [18].

TABLE III: Signature generation and verification times (in ms).

Security level	80	112	128
Signature generation time			
RSA	2.995	14.650	38.585
ECDSA	1.585	2.809	2.835
Paterson	17.316	59.288	123.116
Hess	24.889	91.835	204.757
Barreto et al.	6.715	22.106	57.677
Signature verification time			
RSA	0.113	0.329	0.623
ECDSA	1.884	3.352	3.363
Paterson	24.135	106.804	264.832
Hess	13.232	60.985	155.714
Barreto et al.	33.791	119.080	313.702
Encryption time			
BF	13.157	51.919	122.760
BB	18.031	62.644	129.770
Chen et al.	7.044	23.128	47.845
Decryption time			
BF	6.937	29.730	75.844
BB	13.020	58.707	149.972
Chen et al.	6.959	30.095	75.357
Pairing computation time			
Type A	6.097	28.890	74.265
Type D	13.653	50.429	136.738

Our implementation is based on the PBC library for IBS and IBE schemes, and on the OpenSSL library for the RSA and ECDSA algorithms. Times for RSA and ECDSA are just given as an information, not for an accurate comparison. We make use of two different pairing functions. The first type of pairing is symmetric (it is referred to as Type A pairing in [19]). The second type is asymmetric and is used into Barreto et al. signature (it is referred to as Type D pairing in [19]). 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 summarizes the comparison results (in milliseconds).

From Table III, it is clear that pairing based signatures and encryption algorithms are slower than RSA and ECDSA. This is due to today's available pairing functions that are more complex. Their computation takes more time than an exponentiation in a multiplicative subgroup or a scalar multiplication in a subgroup of elliptic curve.

We notice from Table I that Barreto et al. signature and Chen et al. encryption should be respectively the fastest signature and encryption schemes, as they contain only one pairing function for the signature verification and decryption phases. Their performance efficiency is due to the use of the Sakai-Kasahara key construction scheme, i.e. the private key is computed as: $Priv_{ID} = \frac{1}{Pub_{ID}+s}P$. From Table III, we can confirm, through practical measurements, that Chen et al. encryption is the fastest ID-Based encryption algorithms among the studied one. However, Paterson signature verification is faster than Barreto et al. signature verification, although the Barreto et al. verification needs one pairing operation while Paterson verification needs three. This result is explained by Barreto et al. signature using the asymmetric type D pairing which

TABLE IV: Times for certificate generation and request (in ms).

STA key	key_gen	cert_req	CA_cert_gen
RSA-1024	96.809	2.462	15.205
RSA-2048	650.728	12.874	15.490
RSA-3072	2336.758	34.984	15.744
EC-160	4.840	1.829	18.441
EC-224	7.902	3.038	18.479
EC-256	7.889	3.066	18.500

TABLE V: Time for PKG initialization and public element generation (in ms).

Sec_level	Pairing type	Pairing_gen	PE_gen	Key_gen
80	Type A	6.097	28.677	18.044
112	Type A	28.890	116.299	91.643
128	Type A	74.265	295.527	249.392
80	Type D	13.653	175.589	1.820
112	Type D	50.429	551.537	7.160
128	Type D	136.738	1453.174	18.217

is slower than the symmetric type A pairing used by Paterson, as given in the last 2 lines of Table III.

We conclude from Table III that IBS or IBE time performances mostly depend on the number and type of pairing functions in use. Thanks to the work done by Beauchat et al. [20] leading to defining the fastest existing pairing in less than 1 millisecond, we are confident that very efficient IBC schemes will emerge in the next few years.

C. Benefits of not using certificates and CA

In this section, we first study the different generation times related to the creation of certificates and to the generation of IBC public elements. Then, we show how IBC increases the storage capacity of STA unlikely to certificates.

For better highlighting interest for IBC against RSA/EC schemes in wireless networks, we start by presenting some results in Table IV related to public key certificate creation using the OpenSSL library. We evaluate the time for STA to generate its key pair (key_gen) and to make a request for a certificate from the CA (cert_req), and also the time needed for CA to accept or reject the request and to issue a certificate (CA_cert_gen). CA is assumed to have a 2048 RSA key to sign the certificates. During the simulation, we used three elliptic curves that are referred to as 'NID_secpxxxk1' in OpenSSL.

Table V gives time results for the STA to get IBC keys. These times include the times for generating the pairing function (Pairing_gen), the public elements (PE_gen) and the key (Key_gen). Our simulations are considering the public elements defined by Paterson in [11] and by Barreto et al. in [15] as they use different pairing functions.

Note that using IBC instead of CA and certificates enables saving for each STA the time for generating a certificate request. From Tables IV and V, we can deduce that generation of pairing functions and public elements can take lot of time, especially for higher level of security.

TABLE VI: Certificate size (in bytes).

STA key	cert_size
RSA-1024	1046
RSA-2048	1224
RSA-3072	1399
EC-160	1062
EC-224	1131
EC-256	1164

TABLE VII: Public element size (in bytes).

Pairing type	$\mathbb{F}_p$_size	$\mathbb{G}_1$_size	sec_level	PE_size
Type A	512	160	80	1303
Type A	1024	224	112	2534
Type A	1536	256	128	3767
Type D	175	167	$\geq$80	2184
Type D	347	332	$\geq$112	4219
Type D	522	514	$\geq$128	6254

However, public elements can be generated offline thus saving time at the PKG.

The key generation time is smaller for IBC than for RSA, but is of the same order or at least 25% slower than EC depending on the type of the pairing function in use. For example, Paterson public elements make the key generation time slower than for EC as the Paterson scheme requires the complex operation of hashing the identity to a point of an elliptic curve, and it refers to the private and public keys as two points of an elliptic curve. Also, the Barreto et al. scheme gives similar key generation times than EC keys because the key generation requires the multiplication of an elliptic curve point by a scalar, and only one of the public or private keys is a point of an elliptic curve.

Let's move to storage capacities evaluation. It is clear that IBC selection enables STA to save memory space as the same public elements serve for all the STAs, unlikely to the CA deployment where each STA needs to store the certificates of its peers and the corresponding Certificate Revocation List (CRL). Our simulations with OpenSSL give the size (in bytes) of RSA or EC key certificates in Table VI, and the size (in bytes) of the public elements (PE_size) according to the pairing function type in Table VII. The order sizes of $\mathbb{F}_p$ ($\mathbb{F}_p$_size) and $\mathbb{G}_1$ ($\mathbb{G}_1$_size) are expressed in bits.

A direct comparison between Table VI and Table VII shows that one certificate takes less space than the public elements in a STA memory. However, in practice, a network contains more than 1 STA, and for a fair comparison, we need considering a network of N STAs, with each STA communicating to the other $N-1$. If certificates are used, each STA has to store $(N+1) \times cert\_size + CRL\_size$. That is, each STA has to store $N-1$ certificates corresponding to its peers, the CA certificate and its own certificate. However, if EAP–KERIBA is used, each STA_j has to store only $Pwd_j\_size + PE\_size + token_j\_size + \sum_{i=1,i\neq j}^{N-1} \times (ID_{STA_i}\_size + P_{STA_i}\_size + L_i\_size)$. That is, STA_j stores its own password (shared with the AS), its own token, the public elements and the information recovered from the tokens of the successfully authenticated STA_i, i.e. the identity of its peer ID_{STA_i}, its p ublic

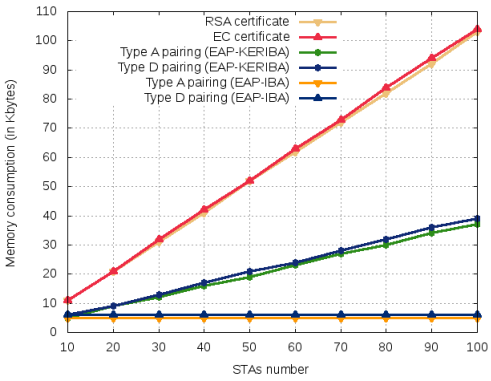

Fig. 4: Memory consumption according to the number of users (security level=80 bits).

point P_{STA_i} and the lifetime of its corresponding private key L_i. If EAP–IBA is used, only the public elements are stored by each STA. This results in an important gain of memory space. Based on the previous formulas, we next evaluate the memory consumption of a STA according to the number of STAs (N) in the network, or according to the level of security. We can deduce from Figure 4 that memory consumption for a fixed 80 bits security level is smaller for EAP–KERIBA than for RSA or EC certificates. However, the memory consumption when EAP–IBA is used is smaller than the one for EAP–KERIBA. But this result is normal because, we do not generate tokens in EAP–IBA contrarily to EAP–KERIBA. Similar results can be deduced for various security levels and a fixed number of peer STAs.

As such, EAP–IBA and EAP–KERIBA methods consume less memory than certificate based scheme. They also reduce the bandwidth consumption as there is no need to send certificate requests and responses including a large certificate. We conclude that our proposed authentication methods are more suitable than classical PKI for networks where STA has memory constraints and good computation capacities.

V. Conclusion

In this paper, we presented two new EAP authentication methods relying on IBC. We informally discussed the security limits of this protocol. We are actually working on the formal verification of EAP–IBA and EAP–KERIBA with the ProVerif tool and we hope to publish our verification results as soon as possible. In addition, the article presented some implementation results leading to the conclusion that IBC is of great interest for wireless networks where storage capacities within STAs are limited.

Furthermore, through implementations and testing, ID-Based signature generation and verification are shown to be at least 25% slower than RSA or ECDSA. However, we are confident that fastest pairing functions will emerge in the future, thus making the ID-Based cryptography and signature scheme adequate for use in any types of networks.

References

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

[2] *IEEE Std 802.11i-2004 - IEEE Standard for Local and Metropolitan Area Networks - Specific requirements-Part 11-Amendment 6: Medium Access Control (MAC) Security Enhancements*, IEEE Standard, 2007.

[3] *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.

[4] 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.

[5] 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.

[6] J. Silverman, *The arithmetic of elliptic curves*, ser. Graduate texts in mathematics. Springer, 2009.

[7] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and communications security*. New York, NY, USA: ACM, 1993, pp. 62–73.

[8] 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.

[9] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Applied Mathematics*, vol. 156, no. 16, 2008, applications of Algebra to Cryptography.

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

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

[12] 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.

[13] 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.

[14] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," vol. 196, 1985.

[15] 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*, B. Roy, Ed., vol. 3788. Springer Berlin / Heidelberg, 2005, pp. 515–532.

[16] 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.

[17] 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.

[18] L. Chen, Z. Cheng, J. Malone-Lee, and N. Smart, "Efficient id-kem based on the sakai-kasahara key construction," 2006.

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

[20] J.-L. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya, "High-speed software implementation of the optimal ate pairing over barreto-naehrig curves," in *Proceedings of the 4th international conference on Pairing-based cryptography*. Berlin, Heidelberg: Springer-Verlag, 2010.