AAWSOME

AAWSOME

Anonymous Authentication for WiFi and some real world scenarios

Abdullah Rasool
s4350693

Gergely Alpár
Supervisor

Radboud University, Nijmegen

> I have not failed. I’ve just found 10,000 ways that won’t work.
>
> *Thomas A. Edison*

CONTENTS

ACKNOWLEDGMENTS

I would like to thank the following people for their support, feedback and hints during the process of writing my thesis. In alphabetical order:

Abdul Ahad Rasool (father),
Dagmar van Ham,
Gergely Alpar,
Hoyhay Asmar (mother),
Koen van Ingen,
Matthias Ghering,
Pol van Aubel,
Thijs Broenink,
Timothy Kuhn,
Wouter Lueks

CHAPTER 1

INTRODUCTION

Due to the increasing deployment of access points for wireless public networks and the developing momentum of portable computing devices, people nowadays have increasingly more access to the internet. In contrast to the early 1990's not only businessmen, but also frequent travellers and people who are grocery shopping use these public hotspots. According to a 2015 industry report published by the Wireless Broadband Alliance [6], whose members include major operators such as Time Warner Company, BT, Google and Intel, said 90 % of WiFi hotspots will be offered by mobile providers 2020. They are used to offload busy mobile networks, provide value-added services. Operators may offer free WiFi for their customers in large cities.

While wireless hotspots provide convenience for the users, these access points also compromises the user's privacy. Daniel Solove develops a taxonomy, see figure 1.1, in which he mentions and classifies threats for users' privacy [47]. In our context the data subject is the user who is authenticated and uses the internet. His network traffic could be collected by other users or by the access point. This can be achieved by looking at identifying aspects such as usernames or MAC-addresses and storing all the traffic related to a user. The collected data could later be processed to create a user profile by aggregation. Furthermore, this data could also be sold to other parties and used for other purposes. A user might also get blackmailed if some sensitive data is found or it might even get disclosed.

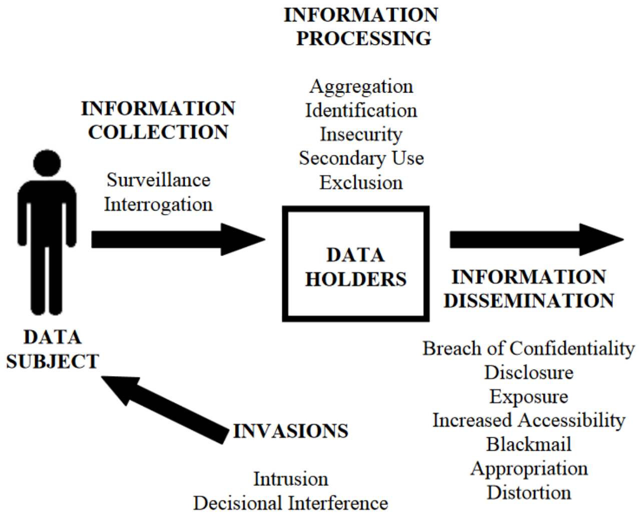

Figure 1.1 Solove's taxonomy [47]

Authentication for WiFi hotspots is important: without proper authentication it is possible for anyone to eavesdrop or interpret the ongoing traffic. WiFi networks without authentication are called open-systems. They can be used for attacking another computer by a distributed denial-of-service attack, or to form a botnet [35]. There are several protection mechanisms for wireless networks: WEP, WPA or WPA2. WEP has proven to be vulnerable [48], its replacement WPA fixed some of the vulnerabilities. Since 2004 the concept of Robust Security Network Association (RSNA) has been used, and is implemented in the 802.1X authentication standard. These techniques are widely used but as it is suggested in [22], some technical measures should be developed to help protect the users' privacy. Attribute-based credentials provides such a mechanism; in the first place, it provides strong authentication by using advanced cryptographic mechanisms. Additionally, it makes use of attributes instead of identities resulting in data-minimisation, which is positive from a privacy point of view. By using attributes a form of anonymous authentication is achieved. An example of an attribute for using a public hotspot that allows travelling users to go online is '*is passenger of flight number X*', or '*is customer of X*' when the access point only allows users who are paying for some services. Creating or modifying such policies upon which internet access is granted are tasks of the hosts of these networks. Moreover, attribute-based credentials also enhance usability [13], since it is possible to automate the authentication process. This was not the case with traditional authentication methods that require remembering and entering username and password combinations. Attribute-based credentials will be discussed more elaborate in chapter 2.

1.1 Contributions

In this thesis we give a thorough analysis of attribute-based credentials and WiFi authentication. We present a protocol that uses attribute-based authentication in the context of public WiFi networks. Finally, a proof-of-concept implementation is presented and some scenarios in which it can be used. This implementation is based on the Extensible Authentication Protocol, it is an authentication framework used in the 802.1X standard. The implementation uses an external library that has implemented all the cryptographic calculations involved in attribute-based credentials. The implementation is running on an access point, it checks if users have the correct attributes required for accessing the network. More details on the implementation are given in chapter 4.

The source code is open source for others to view/modify or deploy it on their own servers and access points. It could also be used for extending other standards to support attribute-based authentication.

1.2 Related work

Direct Anonymous Attestation (DAA). DAA is an anonymous digital signature scheme that aims to provide both signer authentication and privacy [21]. It is used for remote authentication of a hardware module, while preserving the privacy of the user. As in the case of attribute-based credential, one authenticates by means of credentials. DAA is also used to authenticate in a privacy-friendly way in the context of WiFi networks [53]. However, authentication using DAA is computationally expensive and requires a dedicated security chip, a so-called Trusted Platform Module, which is not present in most devices. Most operating systems do not even use this chip, therefore DAA cannot be used on these systems. Since our implementation does not require specific hardware it can be widely deployed. DAA uses hardware security features to prevent credential sharing. But as we will see in chapter 2 this is not necessary since the credentials are linked to a user's private key and therefore cannot be shared [50].

Attribute-based credentials are also used for patients and physicians to authenticate in a medical context [31]. Patients authenticate in a privacy-friendly fashion to their physicians before starting a treatment. This is very fortunate for patients who do not feel comfortable with their disease and decreases the number of wrong medial procedures. Another application is to allow for privacy-friendly authentication for internet-of-things devices [26]. Authentication is necessary since many devices from various contexts need to communicate. However, if there is an identifier attached then people become traceable. By using attribute-based authentication this is no longer a concern.

Attribute-based signatures makes use of attribute-based credentials [37]. It allows a signer who possesses a set of attributes to sign a message with a predicate that is satisfied by the attributes. For example, a doctor who wants to remain anonymous signs a message with the attribute stating I am a cardiologist. This convinces others that he/she is a doctor with a specialism in cardiology without revealing anything else.

As was mentioned in the start of this chapter, a lot of research has been done with respect to the security of WiFi networks and the authentication standards used [36, 15, 22, 35]. But there is far fewer research towards privacy in WiFi authentication. In this thesis we focus on this part, to preserve more anonymity in WiFi authentication.

1.3 Overview

This thesis is divided in the following chapters:

- Chapter 2 gives a theoretical background on attribute-based credentials and one of the implementations, called Idemix. Moreover, the WiFi protocol is briefly discussed and current authentication techniques used in WiFi are discussed.

- Chapter 3 presents the design for attribute-based WiFi authentication. It describes what messages are being sent between the parties and how the modified scheme looks.

- Chapter 4 discusses the technical details including some implementation details, decisions and design choices are explained.

- Chapter 5 shows possible applications and scenarios for which attribute-based authentication could be used.

- At last the conclusions and some further research ideas are discussed in this chapter.

CHAPTER 2

TECHNICAL BACKGROUND

In this chapter we discuss the theoretical background of attribute-based credentials (ABC's) in detail. This includes the ideas behind attribute-based credentials and the underlying mathematical systems. Furthermore, we take a look at the current WiFi standard. Additionally, we look at WiFi authentication methods used for public hotspots. These are described in the IEEE 802.1x standard.

2.1 Attribute-based Credentials

2.1.1 The idea of attribute-based credentials

Before we elaborate on the cryptographic details of ABC's, let us have a look why we want to authenticate using attribute-based credentials. Authentication can be seen as the verification of the identity of some party. Most authentication services have some privacy concerns. We take a brief look at Kerberos, because it is a well-respected and widely used authentication protocols in networks [40]. It is a distributed authentication service that allows a user to prove his/her identity to a verifier. Kerberos uses a ticketing system: a user (or client representing a user) requests a ticket per verifier it wishes to authenticate to. These tickets are granted by a centralized server and does not allow an attacker or verifier to impersonate a user. Kerberos assigns identifiers (or identities) to different entities (principals) participating in the protocol. These identities are in the form "principal_name@realm_name". These user identifiers are sent unencrypted during the authentication process. There are special fields in which these identifiers are sent. An eavesdropper can therefore easily obtain a client's real identity and observe which services are being accessed, which violates the user's anonymity. Even if his real identity may remain unknown it is still possible for an attacker to learn behavioral patterns of service access of a specific users. The reason is when a client wants to access a service, it requests a Ticket Granting Ticket (TGT) from the authentication server. Once it got this TGT it can request Service Tickets (STs) with which it is possible to access services. A client typically uses the same Ticket Granting Ticket to access multiple services. As a consequence the eavesdropper can determine that the same user is accessing these services by tracing this TGT. This system therefore does not accomplish service access untraceability [42]. In addition, it is also possible for the centralized system to trace the users by the services they access.

Attribute-based credentials allow for a special kind of authentication, in contrast with 'traditional' authentication methods, ABCs are not necessarily identifying. An attribute could be seen as a name-value pair. These pairs often represent a person's properties. For example, an adult male would have the following attributes: $\mathrm{gender} = \mathrm{male}$ and $\mathrm{age} = \mathrm{over}\ 18$. In the examples gender and age are the names of the attributes and male and age are the corresponding values. In the case of ABCs the values are encoded so they can be treated as integers so arithmetic operations are possible with these values [7]. A credential is a cryptographic container of a user's attributes and is part of a users' identity. A user receives credentials from a credential issuer. An issuer can be an authority such as the one that issues passports or other official legal documents. This trusted party signs credentials, which ensures that a user is not able to modify the attributes in a credential. Changing the attributes invalidates the signature and fails the authentication of the user. In the same way it is not possible for a user to create custom attributes. To prevent users from exchanging credentials with each other, credentials belonging to the same user are united with a private key. This private key is only known by the user's credential device (which may be a smartphone, laptop or a dedicated device). With these principles in mind a verifier can accept one or multiple attributes as a method to authenticate, if it can trust the issuer.

A credential usually consists of the following items as depicted in figure 2.1:

- The name of a credential;
- The set of attributes consisting of name-value pairs;
- The secret key of the corresponding user;
- The issuer's name and signature.

With attribute-based authentication (ABA) the user does not need tickets, but rather certain attributes. For example if you want to connect to the WiFi network at a university you must possess an attribute stating that you are a student or a member of the academic staff. If we compare this with services like Kerberos we can see that there are no identifying aspects in this authentication process. For example, all students have the attribute 'is student', but only one has the unique identity "clairedunphy@ru.nl". This makes ABA more privacy friendly than traditional authentication methods [29].

In the authentication process the verifier does not receive an entire or more credentials as in figure 2.1. The verifier's policy describes what attributes are required. It is then up to

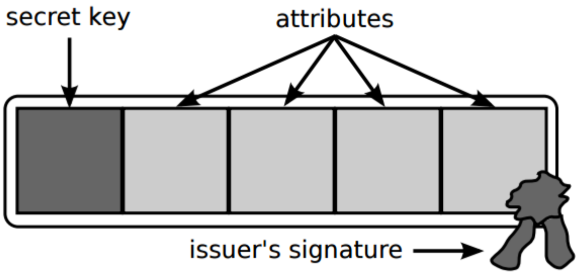

Figure 2.1 Issued credential [49]

the user which credentials and what attributes in a credential he or she wishes to disclose. The user however needs to prove that he or she has all the other undisclosed attributes remaining in a credential. This procedure is called selective disclosure and makes the process more privacy friendly [49].

During authentication a verifier receives the following information from a user:

- The required attributes;
- A proof that the credential containing these attributes is indeed in the possession of the user.

Selective disclosure is discussed in more detail in section 2.1.13.

2.1.2 Underlying cryptography

To understand the technical details of ABCs, we need some overview of their underlying cryptographic systems. Attribute-based credentials are built on top of two hard problems: the strong RSA problem (section 2.1.6) and the representation problem (section 2.1.7). In order to understand why these problems are hard, we must first take a look at some smaller assumptions. We start with a brief discussion about multiplicative groups.

2.1.3 Multiplicative Group

In order to illustrate the cryptographic problems we need to set up a ‘system’ for which it is possible to show why these problems are hard. The system used for the discrete logarithm problem consists of a multiplicative group G. G is constructed by taking a subgroup of Z_p^*. Z_p^* consists of a set of integers: $Z_p^* = \{1, 2, \ldots, p-1\}$. It has $p-1$ elements, which is called the order of the group. The order of G is denoted by q and is smaller than $p-1$, because G is a subgroup of Z_p^* and therefore has a smaller number of elements than the original group. According to Lagrange’s theorem if G is a subgroup of Z_p^*, then the order of G divides the order of Z_p^* [8]. The system furthermore consists of a generator g and a prime p.

let us look at a concrete example: Consider a system $g = 2$ and $p = 11$ so we have $Z_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. This group has order 10. The elements which are generated by the group with generator g are:

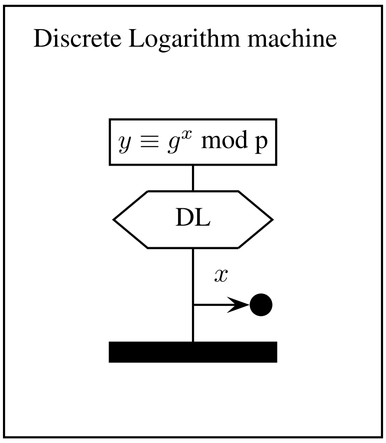

Figure 2.2 The discrete logarithm problem

$$\begin{array}{ll}
g = 2 & g^7 \equiv 7 \bmod 11 \\
g^2 = 4 & g^8 \equiv 3 \bmod 11 \\
g^3 = 8 & g^9 \equiv 6 \bmod 11 \\
g^4 \equiv 5 \bmod 11 & g^{10} \equiv 1 \bmod 11 \\
g^5 \equiv 10 \bmod 11 & g^{11} \equiv 2 \bmod 11 \\
g^6 \equiv 9 \bmod 11 & g^{12} \equiv ...
\end{array}$$

The generator g generates all elements of the original group and it is also noticeable that $g^{12} \equiv g^2 \bmod p$ and so on. Therefore Z_{11}^* is a cyclic group. The somewhat random relation between the exponent i and the generated element $2^i \bmod 11$ is used for the cryptographic assumptions we see next. Not all generators generate all the elements of Z_p^*, if we take $g = 3$ it generates $H = \{1, 3, 4, 5, 9\}$. So H has order 5, which divides 10 in accordance with the Lagrange theorem and therefore H is also a subgroup of Z_p^*.

2.1.4 Discrete logarithm problem

The discrete logarithm problem (DL) can be described as such a system (p, g, q) where p is the order of the group and q is the order of a subgroup generated by g. Consider the group Z_{47}^* which has order 46. According to the Lagrange theorem the subgroups can have orders 46, 23, 2 or 1. If we choose generator $g = 2$, it generates a subgroup with 23 elements. The discrete logarithm problem is as follows: given a system (p, g, q) and y, find x such that $g^x \equiv y \bmod p$, where y is in the subgroup as illustrated in figure 2.2. For example $2^x \equiv 36 \bmod 47$. So the goal is to find the exponent, which is only possible by brute-forcing (ie. trying all possibilities). By using sufficiently large groups this operation is computationally infeasible. More detail on the discrete logarithm problem can be found in [38].

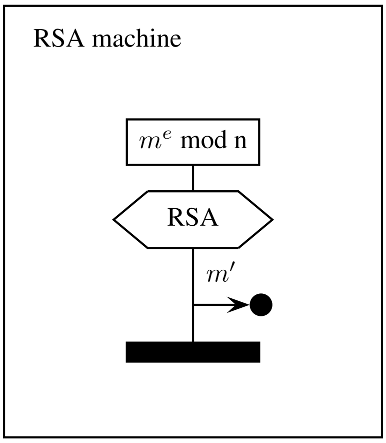

Figure 2.3 The RSA Problem

2.1.5 RSA Problem

An RSA system differs from the system which we have seen thus far. An RSA system consists of $n = pq$ where n is the product of two large primes of the same size. Let e, d be two integers satisfying $ed = 1 \bmod \phi(n)$ where $\phi(n) = (p-1)(q-1)$ which is the order of the multiplicative group Z_n^*. We call n the RSA modulus, e the encryption exponent and d the decryption exponent. The pair (n, e) is the public key and p, q the private or secret key [12].

The RSA problem is not about finding the exponent but rather the base. In the RSA cryptosystem encryption on a message $m \in Z_n^*$ is achieved by using the public key of the recipient of the encrypted message. Encryption is done by raising the message m to e modulo n: $m^e \bmod n$. The exponent is public, so the RSA problem is to find m' such that $m'^e \equiv C \bmod n$ given n and $C = m^e \bmod n$. This problem is believed to be hard as long as the prime factors p, q are hidden [44, 12]. A graphical representation of the RSA problem is presented in figure 2.3.

2.1.6 Strong RSA Problem

The strong RSA problem can be viewed as a combination of the RSA problem and the discrete logarithm problem. In this problem it is hard to find a pair m', e' given a ciphertext $C = m^e \bmod n$ and modulus n such that $C \equiv m'^{e'} \bmod n$ [16]. There are several possibilities for c, d because of the modular arithmetic, for example $5^3 \equiv 3^3 \bmod 7$. You are allowed to choose m, e so it looks easier, but it is called the strong RSA problem because we expect a lot of this assumption. There are a lot of advanced cryptographic systems build relying on the strong RSA problem. A graphical representation of the strong RSA problem is shown in figure 2.4.

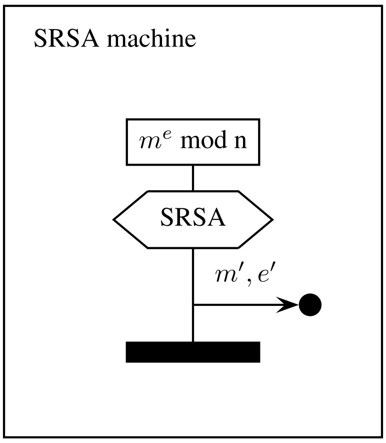

Figure 2.4 Strong RSA problem.

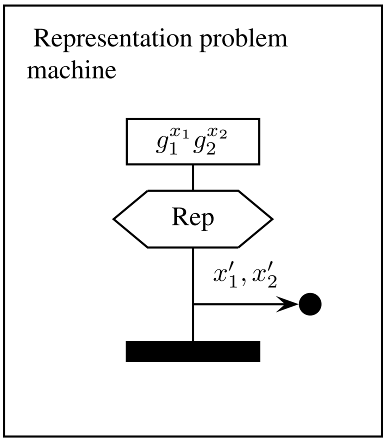

Figure 2.5 Representation problem with two generators.

2.1.7 Representation Problem

The discrete logarithm problem can be generalised to the representation problem. In the representation problem several generators are given instead of one. Consider a set of generators $\{g_1, \ldots, g_L\}$ and a set of exponents $\{x_1, \ldots, x_L\}$ such that $H = \prod_{i=1}^{L} g_i^{x_i}$. The set of exponents is called the *representation* of H with respect to the generators [7]. The goal of the representation problem is to find a set of exponents $\{x_1', \ldots, x_L'\}$, in other words to find another representation, where $H = \prod_{i=1}^{L} g_i^{x_i'}$ holds. In figure 2.5 the representation problem is graphically shown for two generators g_1, g_2 and two exponents x_1, x_2. Unlike the regular RSA assumption the strong RSA assumption can be used in combination with the representation problem because it allows to change the exponents [28].

2.1.8 Pedersen commitment

Commitments form an important class of primitives in cryptography. They are used as a way of flipping fair coins between two players. With fair coins one player flips a coin and he cannot lie about what the outcome was. The player commits to the value of the coin. Furthermore, it plays a crucial part in zero-knowledge proofs [33], which is discussed later in this chapter. In a commitment scheme, a *committer* chooses an element m from a finite set M and releases some information about that value through a commit protocol to a *receiver*. At a later point, the committer may release more information to the receiver to open his commitment. This way the receiver learns m. The basic properties of a commitment scheme are [24]:

- Hiding: a cheating receiver cannot learn m from the commitment.
- Binding: A cheating committer can not change the value of m. The verifier/receiver can check in the opening phase that the value opened was what the committer had in mind originally.

We can use the representation problem to create the Pedersen commitment. For example, we have a message a which we would like to commit to and later reveal. Again we have a system (p, g, q) and choose another generator $h \in G$ where G is the subgroup of Z_p^* generated by g.

Furthermore, we choose a random value $r \in [0, \dots, q-1]$. The commitment we send to the receiver is $C(a) = g^a \cdot h^r \bmod p$. This scheme hides the value of a perfectly, because for any a' there exists an r' where $C(a) = g^{a'} \cdot h^{r'}$ [41, 7] as we saw with the representation problem. The binding property guarantees only that a sender who has limited computational power cannot find a commitment which can be opened in two possible ways [25].

The Pedersen commitment can be generalized to many messages and therefore many generators. However, it always needs to have one additional generator which forms the base of the random exponent which is used for the perfect hiding property. The generalized Pedersen commitment looks like: $C(a_1, \dots, a_L) = g^r \cdot \prod_{i=1}^{L} g_i^{a_i}$, where r is the randomly chosen exponent and $(a_1, \dots, a_L)$ are the messages to commit to.

In the opening phase of the Pedersen commitment the committer shows the values of the random r and the committed message a. In the case of multiple messages the committer reveals all of the messages $(a_1, \dots, a_L)$. This way the receiver can recompute the commitment C and verify whether the committed values were correct [41].

2.1.9 Zero-knowledge proof of knowledge

Zero-knowledge proof systems are typically used as subprotocols within larger cryptographic protocols. They allow a prover to convince a verifier that she holds information satisfying some desirable properties without revealing anything else. These proof systems should satisfy [10, 27]:

- Completeness: the prover can convince the verifier that a true statement is indeed true.
- Soundness: the prover is not able to convince the verifier that a false statement is true.

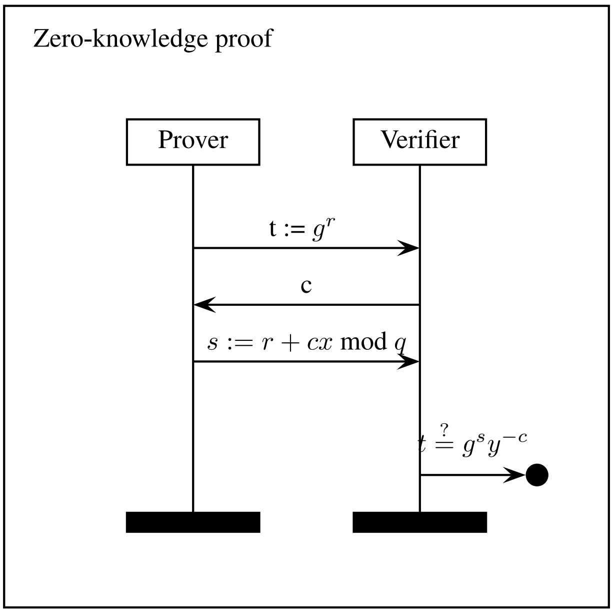

Figure 2.6 Graphical representation of a zero-knowledge proof.

- Zero-knowledge: guarantees that no information leaks and thus protects the privacy of the prover's extra knowledge.

The concept of a proof of knowledge is used in attribute-based authentication. First it is good to get familiar with the notation used. A proof of knowledge is denoted with the letters PK followed by Greek and Latin letters: the former is used to denote variables that are only known to the prover, the latter to mark variables known to both parties.

Schnorr's proof of knowledge of a discrete logarithm can be denoted as $PK\{\chi \mid y = g^{\chi}\}$ and is a simple case of a proof of knowledge. Here we assume that the group is fixed. The prover wishes to prove knowledge of the hidden value χ without disclosing it to the verifier. A proof of knowledge consists of four steps, graphically shown in figure 2.6:

1. **Commitment:** The prover chooses a random r and computes $t := g^r$
2. **Challenge** The verifier generates a random c after receiving the commitment of the prover
3. **Response** The prover sends $s := r + cx$ mod q
4. **Verification** Finally, the verifier checks whether $t \overset{?}{=} g^s y^{-c}$. If it is, the proof is correct and accepted, otherwise it is rejected.

We can show that the verification step indeed is correct by rewriting it:

$$g^s y^{-c} = g^{r+cx}(g^x)^{-c} = g^r g^{cx} g^{-cx} = \frac{g^r g^{cx}}{g^{cx}} = g^r = t. \tag{2.1}$$

More details on zero knowledge proofs can be found in [16]. There exists an adaptation of this protocol, making it a signature scheme. This is based on the Fiat-Shamir heuristic,

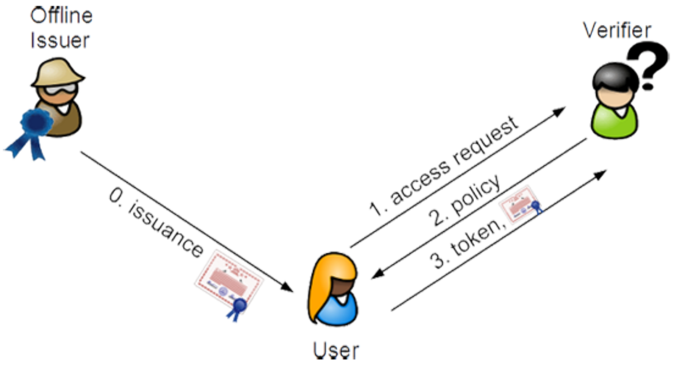

Figure 2.7 The steps in an attribute based authentication environment [2]

it is a popular transformation since it yields efficient signature schemes. In Schnorr's signature scheme instead of a verifier, the prover calls a hash function $\mathcal{H}$ and computes the challenge as $c = \mathcal{H}(t||m), r$, where m is the message that is signed, t is the commitment and r is the random exponent as was shown in figure 2.6. The signature is the pair (c, s) where $s := r + xc$ mod q. Verifying the signature entails computing $t' := g^s \cdot y^{-c}$ mod q and checking whether $c \overset{?}{=} \mathcal{H}(t'||m)$ holds [46]. In our implementation we use the notion of this non-interactive zero knowledge proof of knowledge to convince the verifier that we have the remainder of the attributes in a credential.

2.1.10 Idemix algorithm

After having gone through all the cryptographic building blocks we can take a look at the algorithm which is used for attribute-based credentials. There are two well-known algorithms which can be used for ABCs: U-Prove [14] and Idemix [18]. In this research we focus on Idemix. U-prove relies on Brands' signature and one downside of it is, generated signatures can not be randomized. This means that it is possible for a credential verifier to track when and how many times a certain user authenticates, because it can store the signature and the time of authentication. In Idemix it is possible to randomize signatures which makes unlinkability (see section 2.1.13) possible.

Idemix is designed by IBM Research in Zürich and relies on the Camenisch-Lysyanskaya signature scheme [18] which can also be applied on the Pedersen commitment. This scheme allows for the use of zero-knowledge proofs. Figure 2.7 shows a graphical representation of the steps for authenticating using ABCs. There are three parties involved: the issuer, the verifier and the user. First, the user registers with the issuer and it generates a credential for the user. This is called the registration or issuing protocol and is discussed in section 2.1.12. Prior to the issuance protocol, the issuer needs to set up some cryptographic keys. This is done in the key generation protocol which is discussed next. Finally, the user can authenticate to a verifier using the selective disclosure protocol. The goal of the selective disclosure protocol is to disclose the required attributes to the verifier and prove that the other attributes are in the issued credential [49].

The Idemix algorithm thus consists of three parts:

- Key generation
- Issuing a credential
- Selective disclosure or authentication

2.1.11 Key generation

In this step the credential issuer has to generate its system and keys. First of all, a special RSA modulus $n = pq$ has to be generated such that $p = 2p' + 1, q = 2q' + 1$, where p, p', q and q' are large prime numbers [18]. The quadratic residue group QR_n is defined as all quadratic elements of Z_n^*, that is every element r such that $x^2 \equiv r \bmod n$.

As with other RSA applications the private key is p, q, the factors of the modulus n. The public key is a tuple of elements of $QR_n : (S, Z, R, R_1, \ldots, R_L)$. Finally, the user (user's device) chooses a random secret key a which is kept secret [18].

2.1.12 Issuing a credential

In the issuing protocol an issuer and a user create a new credential. The issuer and user both participate in this protocol and each party has their own input:

- Issuer: has the private key p, q as private input and the public key $(n, Z, R, R_1, \ldots, R_L)$ as public input.
- User: has the private key a and the user's attributes $a_1, \ldots, a_L$ as input.

An Idemix credential is a signature on a commitment, which is the Pedersen commitment on the user's attributes. The issuer does not learn the user's secret key or the final signature [17]. The issuing protocol is an interactive protocol between the issuer and the user and is graphically presented in figure 2.8:

- The user first commits to the secret value a using a Pedersen commitment by computing $U = S^{v'} R^a \bmod n$ where v' is randomly chosen to hide the value of a. In addition does the user prove that he/she knows the value of U with respect to v' and a by sending a proof of knowledge. In other words, she proves having the knowledge of the randomly generated value (v') and her private value (the committed value), without actually revealing them to the issuer.
- Next, the issuer generates a pre-signature. The user generates the actual signature using this pre-signature and v' in the next step. A signature consists of a three tuple (A, e, v). The issuer chooses a random e and v'', and computes A using the formula: $A = \left(\frac{Z}{US^{v''} \prod_1^L R_i^{a_i}}\right)^{1/e} \bmod n$, where a_i is the integer representation of an attribute and R_i is the base of an attribute. Alongside, this pre-signature the issuer sends a proof of knowledge where she proves the knowledge of $\delta = \frac{1}{e} \bmod \phi(n)$. With this proof of knowledge she also indirectly proves knowledge of the private key since $\phi(n) = (p - 1)(q - 1)$ where p, q is the private key.
- After receiving this pre-signature, the user computes the complete signature by calculating the value of v by adding her own generated v' with the issuer generated random

Issuer Secret: p, q	System Public: $n, Z, R, R_1, \ldots, R_L$	User Secret: a User attributes: $(a_1, \ldots, a_L)$
	$\xleftarrow{U,PK_1}$ $PK1 := PK\{(\nu', \alpha) \mid U \equiv S^{\nu'} R^{\alpha} \bmod n\}$	Choose random v' $U := S^{v'} R^{a} \bmod n$
Choose random v'' Choose random prime e Compute $A := \frac{Z}{US^{v''} \prod_1^L R_i^{a_i}}^{1/e} \bmod n$	$\xrightarrow{(A,e,v''),PK_2}$ $PK2 := PK \left\{ (\delta) \mid A \equiv \left(\frac{Z}{US^{v''} \prod_1^L R_i^{a_i}} \right)^{\delta} \bmod n \right\}$	
		Compute $v := v' + v''$

Figure 2.8 Graphical representation of the credential issuing protocol [7].

v''. This way the issuer does not know the final signature because it does not know v'. The signature will now look like the three tuple mentioned earlier (A, e, v).

- The user can check whether the resulting signature is indeed valid $Z \overset{?}{\equiv} A^e S^v R^a \prod_1^L R_i^{a_i} \bmod n$.

2.1.13 Selective disclosure

In the selective disclosure or verification protocol the user may choose not to disclose all attributes in a credential but only those required by the verifier. This is done to make authentication more privacy friendly, since the user may choose if he wishes to disclose more information than required. This also leads to data-minimization. The selective disclosure protocol is split up in two parts: randomization of the signature and prove knowledge of all non-disclosed attributes.

The user first randomizes the signature, by choosing a random r from a large interval. By randomizing a signature the user ‘blocks’ verifiers from tracing the user. A randomized signature looks like (A', e, v') where $A' = A \cdot S^{-r}$ and $v' = v + e \cdot r$ [49]. One can check that this randomized signature is still a valid signature by rewriting the randomized signature 2.2, where $R := R^a \prod_1^L R_i^{a_i}$:

$$A'^e \cdot S^{v'} \cdot R \equiv A^e \cdot S^{-(er)} \cdot S^v \cdot S^{er} \cdot R \equiv A^e \cdot S^v \cdot R \equiv Z \qquad (2.2)$$

Next the user must prove knowledge of all non-disclosed attributes. If the user chooses not to disclose any attribute then she proves knowledge of all attributes, otherwise only of those which it did not disclose. So in the selective disclosure phase a user sends all disclosed attributes, A' from the randomised signature and a proof of knowledge of all undisclosed attributes and other parameters to the verifier [49].

An example showing a proof of knowledge of a non-empty disclosing set, $\mathcal{D}$:
$PK\{(\epsilon, \nu', \alpha, (\alpha_i)_{i \notin \mathcal{D}}) \mid Z \cdot \prod_{i \in \mathcal{D}} R_i^{-a_i} \equiv A'^{\epsilon} S^{v'} R^{\alpha} \prod_{i \notin \mathcal{D}} R_i^{a_i} \text{ (mod n)}\}$

In this proof of knowledge the issuer checks if the signature on the commitment is still valid. If we look at the zero-knowledge proof we can see an equivalence:

$$Z \cdot \prod_{i \in \mathcal{D}} R_i^{-a_i} \equiv A^{'\epsilon} S^{v^{'}} R^{\alpha} \prod_{i \notin \mathcal{D}} R_i^{a_i} \bmod n \tag{2.3}$$

The goal of this proof is to check whether the user knows the representation of the public key Z with respect to the generators. If the user does know this representation (ie. the exponents) then this is a sufficient guarantee that the user owns a credential signature [7]. If the user discloses some attributes, the verifier can reconstruct some part of the product R' by way of $\prod_{i \in \mathcal{D}} R_i^{a_i}$. In the proof of knowledge the user proves having the non-disclosed attributes and together with the disclosed attributes the verifier can check whether the user has a valid credential signature. This is the case when the equivalence in equation 2.3 holds.

2.2 WiFi and WiFi Authentication

In this section we take a look at the 802.11 wireless networking standard. 802.11 is a set of standards which describe the transmission of wireless messages. These standards together define what WiFi is and how it functions. Authentication mechanisms are also defined in 802.11 standards. WEP was the security algorithm used in the original 802.11 standard but it had some vulnerabilities. WPA fixed some of these, and implemented a draft version of the 802.11i standard. Later, when WPA2 was introduced it implemented the complete 802.11i standard. Next, we have the port-based access control mechanism, 802.1X, which we used in our implementation. The 802.1X standard defines the use of the Extensible Authentication Protocol (EAP) over LAN. We use EAP to construct our privacy-friendly authentication mechanism.

2.2.1 The IEEE 802.11 Standard

A brief description of the 802.11 standard is given here, but the author expects that the reader is familiar with how WiFi works.

The 802.11 standard supports two modes of operation: ad hoc and infrastructure mode. In the ad hoc mode two or more wireless stations recognize each other and set up a peer-to-peer connection without using any existing infrastructure. These stations communicate with each other directly and each station broadcasts their presence in the network. In the infrastructure mode there is a fixed entity called an access point (AP) that relays all data between the mobile stations that are *associated* with the access point. An AP and its associated stations together form a Basic Service Set (BSS). They communicate in an unlicensed RF spectrum (2.4 or 5 GHz). A collection of access points can extend a BSS into an Extended Service Set (ESS) via the use of a wired network [39]. Figure 2.9 shows two mobile nodes in each basic service set connected with an access point. The two basic service sets form an extended service set via the wired network.

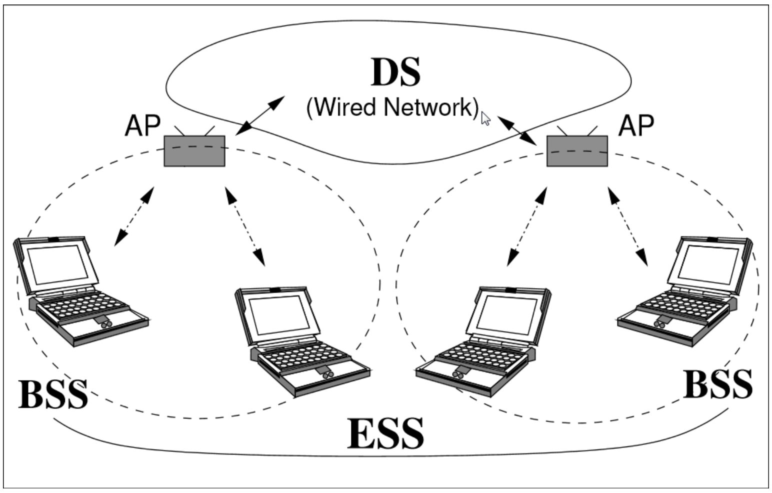

Figure 2.9 Components in 802.11 wireless network [39]

Basic service sets have a range within which they can communicate with their stations. When a mobile station goes beyond the range of one access point and enters another BSS a *handoff* occurs. During the handoff, several messages (management frames) are exchanged between the station and the access point. The access points also exchange some information via the wired network. After the handoff the station is now associated with the other BSS.

Due to the lack of a physical connection between the mobile station and the access point, authentication of the mobile station is an important security requirement. By implementing an authentication mechanism it is possible for the host of a WiFi network to determine who is allowed and who is not. In addition, authentication makes it possible to trace actions back to a specific user [52]. Confidentiality of users is also important since we do not want others to see what we are doing on the internet. The 802.11 standard had some mechanisms for security and authentication:

- Entity authentication, including open system and shared key authentication
- Wired Equivalent Privacy (WEP)

However, both mechanisms have proven to be vulnerable [48]. The objective of the IEEE 802.11i standard was to enhance the vulnerable security aspects of the 802.11 standard. Furthermore, the new standard should introduce key management and establishment mechanisms, define encryption and improve the authentication process. The 802.11i standard incorporates authentication enhancements from the 802.1X security standard [19].

2.2.2 WiFi Authentication: 802.1X Standard

The 802.1X authentication framework works with Robust Security Network Association (RSNA). An RSNA defines a number of security features in addition to WEP and the 802.11 standard: [5]

- Enhanced authentication mechanisms for stations;

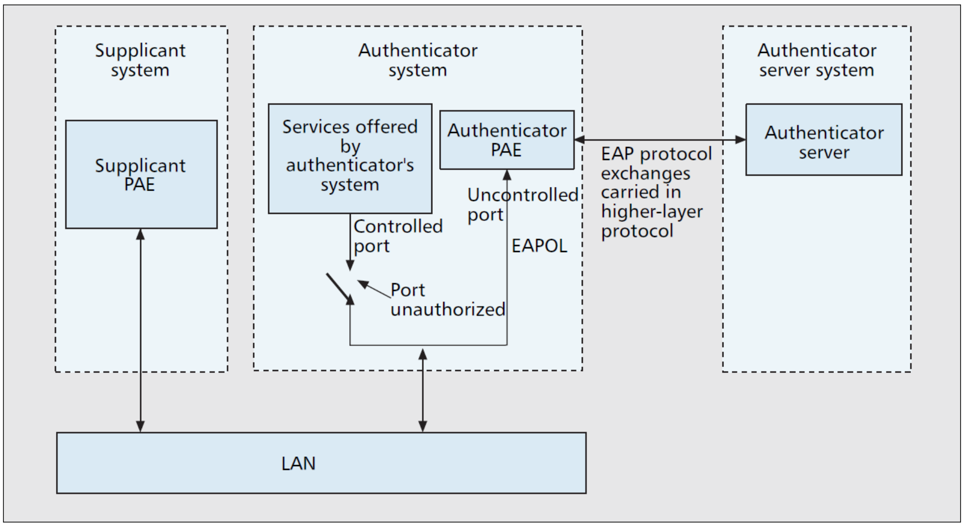

Figure 2.10 Graphical representation of 802.1X system with the three parties [19].

- Key management algorithms;
- Cryptographic key establishment;
- Enhanced data cryptographic encryption mechanisms;
- Enhanced cryptographic encapsulation mechanisms for robust management frames.

There are three parties involved in an 802.1X authentication system:

- Supplicant, which is usually a user requesting access to the wireless network.
- Authenticator, which in 802.11 networks is an access point (AP).
- Authentication server, which handles the authentication for the authenticators, but they could be integrated into the AP.

The supplicant and the authenticator have a port access entity (PAE). The PAE handles the algorithms and protocols associated with the authentication mechanisms. The PAE of the authenticator controls the state of the *controlled port*. In figure 2.10 we can see that the controlled port is unauthorized. This means that the user is not yet authenticated and the authenticator will block all traffic other than 802.1X messages. During authentication the uncontrolled port is used to communicate with the PAE of the supplicant to complete the authentication [19].

The 802.1X standard incorporates the Extensible Authentication Protocol (EAP) to provide a number of authentication schemes. Some examples of supported authentication methods are PAP, MD5, MSCHAP. Authentication using EAP is depicted in figure 2.11 but the number of messages may vary depending on the authentication method used. In some cases the authenticator and the authentication server are the same physical device.

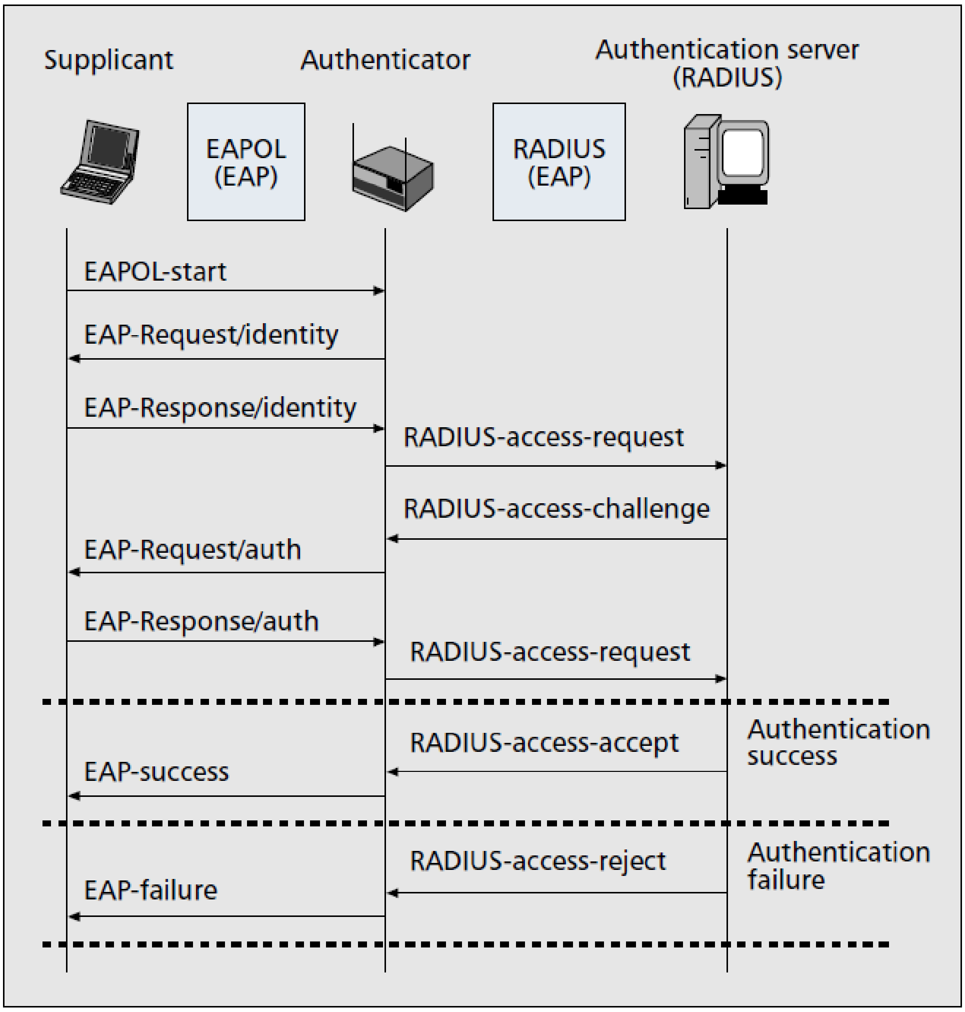

Figure 2.11 Messages in 802.1X authentication process [20]

Before the authentication between a supplicant and the access point is started, the client must first associate with that authenticator. After association the supplicant can start the authentication process, at that point the controlled port is unauthorized. The following messages between the PAE of the parties are sent:

1. The supplicant sends an EAPOL-Start[1] message.

2. After the supplicant receives this message, it responds with an EAP-Request/Identity to obtain the supplicant's identity.

3. The supplicant sends an EAP-Response/Identity message back. This contains the supplicant's identity which might be a username.

4. The authenticator encapsulates the EAP-Response/Identity message in a RADIUS Access-Request message to the authentication server.

5. The authentication server challenges the supplicant by sending an RADIUS Access-Challenge message to the authenticator. The authenticator relays this message to the supplicant as an EAP-Request/Auth message.

6. The PAE of the supplicant then sends an EAP-Response/Auth message which gets relayed to the authentication server by the access point.

   Depending on the authentication scheme there might be more messages exchanged. The authentication server decides when the user should be accepted.

7. If the authentication has succeeded the authentication server sends an RADIUS Access-Accept to the access point. The access point then knows that the supplicant is authenticated and grants the supplicant access to the internet. Furthermore, it sends an EAP-Success message to the supplicant. The authentication process is now completed.

8. If the authentication fails, a RADIUS Access-Reject packet is relayed by the access point as an EAP-Failure message to the user. The status of the controlled port will remain unauthorized.

Usually there are several access points connected to one authentication server. User profiles are then stored in a centralized database to release the burden of the access points saving large amounts of MAC addresses, usernames and passwords. In case of a handoff a user does not have to reauthenticate to an access point utilizing the same authentication server.

At the start of this chapter we discussed Kerberos as an example of a identifying authentication scheme. Most other EAP authentication methods are based on a username/password combination, which results in an identifying authentication method. In the next chapter a new protocol is described which uses attribute-based credentials to make a non-identifying authentication scheme.

[1] Extensible Authentication Protocol Over LAN (Local Area Network)

CHAPTER 3

THE CONCEPT OF ATTRIBUTE-BASED WIFI AUTHENTICATION

In this chapter we present a description for an anonymous authentication protocol for WiFi. First, we discuss the protocol on a high level in which we mention the key components, the ideas and some examples. Next, we link this description to the cryptography discussed in chapter 2. The technical implementation and design choices are going to be discussed in the next chapter.

3.1 High-level protocol

In WiFi hotspots we wish to regulate access to the network. This is achieved by letting requesting users authenticate and only allow those who meet some condition. Authentication and restricted access are important for the users on the network and for their privacy. As we have seen before, if we have no security mechanisms and an attacker is on the network then he can eavesdrop on users' traffic and inject messages into the wireless network. The problem with current authentication mechanisms is that they are identifying. Recall Kerberos from chapter 2, which uses an identifying ticketing system. It is possible for the ticket granting server to link authenticating users across multiple services since they are using the same ticket.

A username and password combination is another authentication method but this requires storing all user credentials into a database, which could become an object of interest for attacks. Furthermore, authenticating with a username is also identifying and therefore not suitable for an anonymous authentication protocol. In addition, passwords are known to be vulnerable to dictionary attacks, phishing, and social-engineering attacks. Both methods are not suitable for an anonymous authentication standard.

As was mentioned before, a credential is a cryptographic container of a user's attributes. An attribute is described by the attribute's name (for example *first name*) and the corresponding value (for example *John*). A credential is created by an issuer and by doing so he vouches for the correctness of the attributes with respect to the user [43]. Later when the user requests a service from a service provider it shows what information is required. The user presents these attributes and gives a proof of the undisclosed attributes [11]. This is called selective disclosure and it allows users to disclose a minimum amount of information called data-minimization.

Our proposal for an anonymous, non-identifying, authentication protocol is to make use of attribute-based credentials. Using this protocol users can authenticate in a privacy-friendly way. There are benefits for the user as well as for the verifier or service provider by using attribute-based authentication [7]:

- Authenticity: the content of a credential is signed and therefore can not be modified.
- Unforgeability: it is not possible for a third party or a user to forge a valid credential.
- Non-repudiation: the issuer could not deny that an issued credential was issued by him.
- Non-transferability: prevents the user to transfer a credential to another user.
- 'Offline' issuer: the issuer is not part of the authentication process.
- Issuer unlinkability: an issuer can not trace his credentials. That is, it is not known if in an issuing and verification protocols the same credential was used.
- Multi-show unlinkability: verifiers cannot trace the activities of the user.
- Unknown data: companies do not have to process personal data, since everything is anonymised. Processing personal data is a high risk for businesses, because it demands proper security measurements for data storage.
- Selective disclosure and data minimisation as we mentioned earlier.

There are some other benefits of using ABCs for authentication. From a business perspective it is now easier to differentiate classes of users. For example passengers traveling first class, having paid more for their ticket, can receive faster WiFi connection on an airplane. A benefit for users might be delegation of attributes. Consider a family, of which all of the members have their own smart-device and want access to the internet. At the time of writing the solution is that one family member buys access for their own device and all others use that one device to do all of their own online activities. One might imagine that this is a rather suboptimal solution. Moreover, every member could buy access for their own device. However, not everybody might own a creditcard, needs to use the internet very long or it might be too expensive. A better solution would be for that one member to buy a credential which grants him internet access and he may also delegate that credential to a fixed number of others [9]. They can now access the internet using their own devices at the same time. The latter solution is also more profitable for businesses hosting WiFi hotspots.

At the moment of writing delegation of credentials is possible but it computationally too expensive.

3.2 Detailed description of the protocol

In this part we take the high-level description of attribute-based authentication and link it with the cryptography discussed in chapter 2. The protocol has two steps: first the user must register in order to receive a credential and thereafter the user can authenticate using that credential.

3.2.1 Registration/Issuance protocol

In the registration phase the user registers at a credential issuer. This party needs to know valid information regarding the users. The credential issuer is trusted for verifying that the attributes in a credential authentically belong to that user [26]. For example if a university states that person A is a student, then other parties can be ensured that person A is indeed a student. This idea is used for the issuance protocol. The credential issuer signs a Pedersen commitment on some user's attributes. A commitment hides and binds the attributes, making sure the values are confidential and are unchangeable (see section 2.1.8). By signing the commitment it is not possible for a user to modify the commitment, since this invalidates the signature. The user then stores this credential on his device, so it remains under his control.

The registration protocol is not part of the WiFi authentication protocol, it is ran prior to the verification protocol. It can be ran when the user is online, with a remote credential issuer, or offline with personal contact. We see examples of the registration protocol in chapter 5.

3.2.2 Verification protocol

In our scenarios the verifier is implemented as the party that regulates access to the internet, which is the authenticator in the WiFi authentication scheme. During the verification protocol the user requests access to the network. This is the anonymous authentication protocol within which attribute-based credentials are used. It differs from other authentication mechanisms used for WiFi networks, since it does not have to identify a user.

In order to make this work we should extend the Extensible Authentication Protocol, or modify an existing EAP authentication scheme. In this extension we define our new protocol. In this protocol the verifier maintains a policy which requesting users must meet. In the next chapter we discuss the technical steps which are taken. An example of a policy could be that all female adults are allowed to the network. The users can show that they meet this policy by sending some credential's attributes. As we saw in the previous chapter this is done by using the selective disclosure protocol. The user sends the required data to an access point:

- Randomised signature;
- Disclosed set of attributes;
- Zero-knowledge proof of knowledge with respect to the undisclosed attributes.

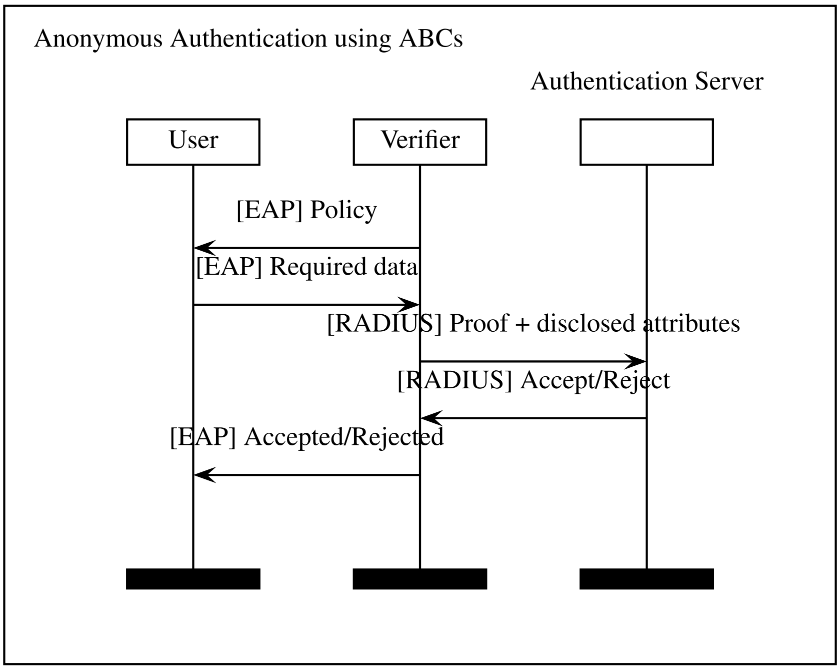

Figure 3.1 High-level graphical representation of attribute-based authentication.

This data is sent in an EAP Response message which then gets verified to check whether the user complies with the policy. If that is the case the user is granted access to the network, otherwise the user is rejected. Figure 3.1 shows a graphical representation of this authentication protocol. The user may choose not to disclose more information than is necessary. That is allowed, because the verifier only needs to know the required attributes. The user must prove possession of the remainder of the attributes. Furthermore, the information which is disclosed does not have to be identifying. The selective disclosure, which leads to data-minimization, and the non-identifying attributes in a credential make this authentication scheme more privacy friendly than existing solutions.

CHAPTER 4

THE IMPLEMENTATION OF ATTRIBUTE-BASED WIFI AUTHENTICATION

In this chapter we discuss the implementation of the anonymous authentication for WiFi as discussed in the previous chapter. We start with a description of the tools and libraries which are used, followed by some unsuccessful attempts. Next, we elaborate on the design and choices for the current implementation. We end this chapter by mentioning possible improvements for the future.

4.1 Open-source applications & libraries

The implementation involves two parties:

- The authenticator (or access point)
- The supplicant (or user requesting access)

In practice there may be several access points connected to a centralized authentication server which takes over the authentication part from the access points. To keep the implementation small and easy, we omit the centralized authentication server and we only have one access point. A WiFi network with one access point could in practice only be used in a very small venue. In order to implement attribute-based WiFi authentication we need to alter current access points and supplicants. Fortunately, these parties are available as open-source applications which we have to modify so they satisfy our requirements.

For the authenticator we use hostAPD which is a user space daemon for access points and authentication servers. It implements the 802.11 access point management, EAP authenticators, EAP server and more [1]. As a counterpart we use WPA Supplicant which is a supplicant daemon that implements key negotiation with a WPA authenticator and controls the authentication and association of the driver [3].

These two applications handle the networking part of the authentication. They construct the EAP messages and send them to each other over the link layer. However, we also need to have a payload for these messages. Recall the selective disclosure step in which we had to send the disclosed attributes and a zero-knowledge proof of knowledge of the undisclosed attributes. For the cryptographic part of the authentication protocol we adopt a third-party library, credentials idemix. This library - developed at the Radboud University - implements all the mathematics of attribute-based authentication. By using high level API calls it is possible to issue a credential, to generate a proof and to verify whether this proof is correct [23].

4.2 Some unsuccessful approaches

In this section we discuss some notable approaches which did not succeed during this project. We also provide an analysis why it did not work and what direction was taken after that approach.

4.2.1 Authentication server

For a first naive approach, we considered to make the supplicant communicate directly with an open-source authentication server (implemented using freeRadius). Once a user requested access and was authenticated correctly nothing would happen (ie. the user would not have internet access). This is because the authentication server did not belong to any authenticator that would grant the user access to the internet after a succesfull authentication.

As it is shown in figure 4.1 we skipped the authenticator which offers the internet service. The supplicant and authentication server exchange EAP messages directly to each other. When the authentication succeeds, the port access entity (PAE) of the supplicant receives a message to change the state of the controlled port to authorized. If the port is in this state, then it is possible for the supplicant to use the service which is protected by the port (ie. the internet). The supplicant however does not have services which it needs to protect, so nothing happens.

After this approach it was obvious that there should be an access point between the user and the authentication server. Direct communication between the user and the authentication server would not work. Instead a structure shown in figure 2.10 is needed.

4.2.2 Card-oriented library

The Radboud University has a special pilot project, IRMA, for developing software used in attribute-based authentication and testing it in some applications[1]. There are several libraries developed by members of this pilot. Prior to using the credentials idemix library,

[1] More information on: https://www.irmacard.org/

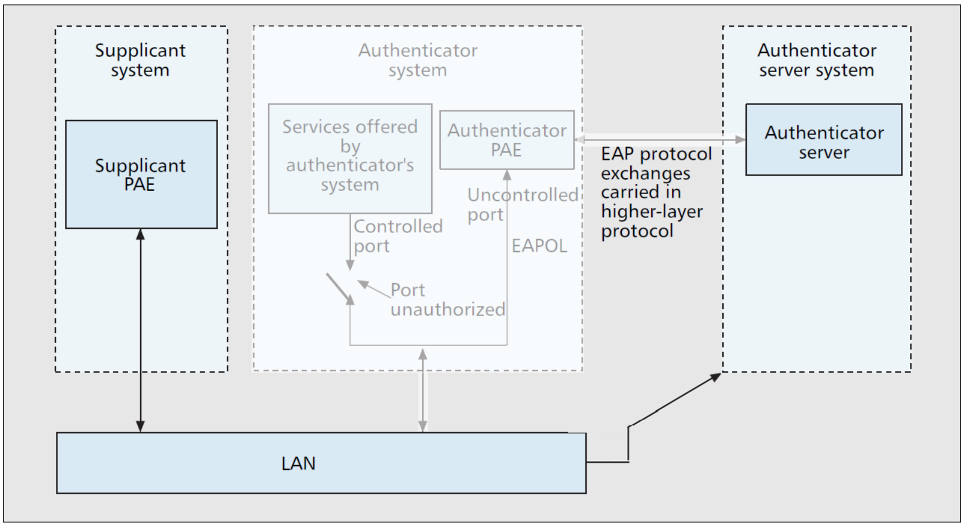

Figure 4.1 Attempt with authentication server.

we used a library called Silvia. This proved to be too much oriented towards smartcards and not suitable for WiFi authentication. Silvia was written in C++ and we thought that it would be possible to call functions written in a C++ library from hostAPD and WPA Supplicant, written in C. This was however very hard to achieve and within the context of the applications used. We dropped this idea and chose to use credentials idemix instead.

4.2.3 Existing access point

Before using software which emulates an access point we attempted to authenticate using WPA Supplicant to a 'real' access point. It was possible to respond to the EAP-Request Identity message by sending an EAP-Response Identity. But when trying to send a longer message than specified in the protocol the access point would drop it. The access point considered this packet as damaged. It should however be able to receive and process long messages, because attributes and the zero-knowledge proof can be seen as very long messages containing numbers upon which some aritmetic operations are performed.

This made it necessary to use a custom access point in which it is possible to disable some sanity checks and process the 'damaged' messages. This would allow the development of a draft implementation of our new authentication protocol. HostAPD allows for these options, since the code is open source it is possible to modify it and use the adapted version. Furthermore, it is possible to have the authentication server incorporated in hostAPD. This made the implementation less complicated and the result more compact.

4.3 Final implementation

The final implementation makes use of an existing authentication method within the Extensible Authentication Protocol (EAP) framework which is no longer used, EAP-MD5.

EAP-MD5 is a password-based network authentication method which contains the following steps [34]:

1. EAP-request identity: the authenticator requests the username of the supplicant and sends along a 1-byte identifier.

2. EAP-response identity: the supplicant sends the username.

3. EAP-request challenge: the authenticator sends a 16-byte random challenge.

4. EAP-response challenge: the supplicant responds with the MD5 hash of the identifier, his password and the challenge.

5. EAP accept/EAP reject: depending on whether the response for the challenge was valid.

EAP-MD5 has proven to be vulnerable to man-in-the-middle attacks: an adversary can act as an access-point and perform the EAP-MD5 steps. It receives the hashed identifier, password and challenge. If the adversary uses a fixed identifier and challenge, it can brute-force until it finds the user's password using a dictionary attack. In addition, the MD5 hash function is also vulnerable [51]. Therefore, finding the user's password using MD5 is rather effortless with enough computational power. Regardless of the vulnerability in MD5, we use EAP-MD5 to send our own messages. We do not use however any weak cryptography related to MD5. Our application aawsome-crypto uses credentials idemix to issue a credential, generate a proof and verify a given proof. Due to a discrepancy between the programming languages of credentials idemix, hostAPD and WPA Supplicant we decided to write these artifacts to a textfile. This could then be read by hostAPD or WPA Supplicant and send to each other over the link layer.

A downside of EAP-MD5 is that is does not generate keying material, so after the supplicant is authenticated the traffic is sent unencrypted. This is dangerous because other clients on the network see the traffic and all sensitive data which may be send. To handle this issue we first create a TLS tunnel through which the authentication is handled and after a user has successfully been authenticated the tunnel is used to send the traffic. This data is encrypted so it is not possible for others to see what is being send.

We can now list the steps which are taken by our authentication scheme:

1. EAP-request identity: the authenticator requests the username of the supplicant and sends together a 1-byte identifier.

2. EAP-response identity: since our authentication scheme is anonymous the supplicant does not have a username. Instead, it sends the required attributes which the authenticator requires. At this point, the authenticator (or verifier) does not send a policy so the user needs to know what attributes are required. If the user does not have the correct/required attributes, the user is rejected.

3. The supplicant and the verifier set up a TLS tunnel. All further messages are encrypted.

4. EAP-request challenge: the authenticator sends a 256-bit unpredictable random challenge and an 80-bit contextstring. The context is used to recognize the verifier, it

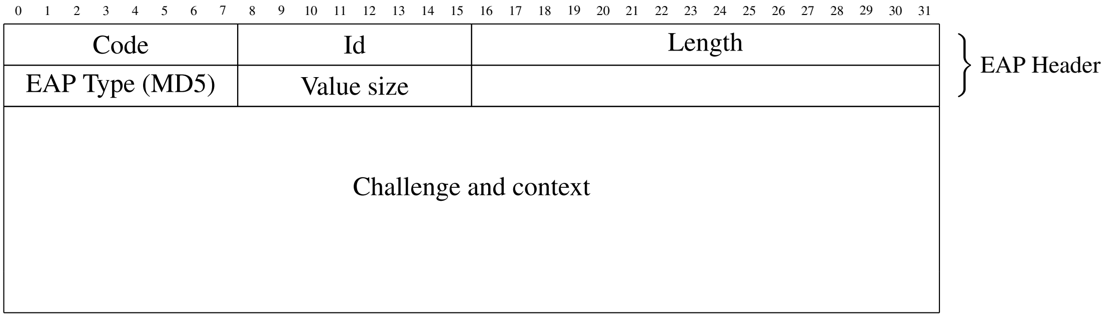

Figure 4.2 EAP-Challenge request packet

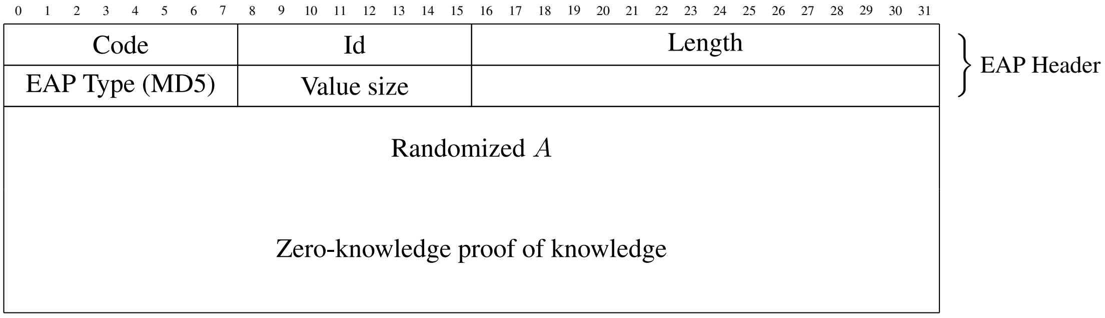

Figure 4.3 EAP-Challenge response packet

contains the encoding of the public key of the verifier. A packet of the request challenge is shown in figure 4.2.

5. EAP-response challenge: it sends part of a randomized signature as was seen in chapter 2 and it sends a zero-knowledge proof of knowledge of the undisclosed attributes. A packet of the response is shown in figure 4.3.
6. Finally, if the proof is verified then the user is authenticated.

Our implementation is a proof of concept which demonstrates that it is possible to send a large message to an authenticator. Compared to other authentication mechanisms we use the full packet size and that was a challenge with respect to the tunneling process. Since our adapted messages form the payload (the data) of the TLS messages and if these TLS messages get too long they are fragmented (ie. they are split in multiple messages). Fragmentation shows strange behavior in execution, so therefore we restructured the response.

During the authentication process there is no overflow of disclosed personal information, only the required attributes are disclosed. Hence, this is a very simple yet functional authentication scheme for anonymous authentication for WiFi hotspots.

4.4 Possible improvements on the current implementation

There are a few drawbacks with our current implementation. First and foremost, it makes use of the messages of EAP-MD5. It is better to make a new custom authentication scheme, with its own EAP-type and packet structure which may be a better fit for the data transferred with this protocol. This new protocol can also generate keying material, therefore not relying on TLS. From a usability perspective there are major improvements to be made: since this version of WPA Supplicant is not yet incorporated in the operating system, the user should block his/her current networking manager or use another wireless interface. Our version of WPA Supplicant should be incorporated, so it would be easier for users to authenticate.

As was mentioned at the start of this chapter, there is no centralized authentication server. Because of time constraints we chose to implement a simplified model. As a next step the implementation should use a centralized authentication server like freeRadius[2] to handle the authentication process. Authenticated users are stored in a centralized server by their MAC addresses. This allows mobile handoffs and is a step towards a real-world implementation. Aditionally, the access points do not have to make complex cryptographic calculations. Our version of hostAPD could also be incorporated in an open-source access point firmware (like openWRT or dd-wrt). This makes the set up more robust than using a Raspberry Pi as was used for this demo implementation.

Finally, from a programming perspective it would be nice to have a cryptographic library, as credentials idemix, with the same (or maybe higher) level API's and written in the same programming language as hostAPD and WPA Supplicant. The method of writing intermediate results to a textfile is suboptimal and prone to concurrency issues when multiple users are requesting access since it may overwrite session files. This may also result in security issues, because some users may overwrite files and get access to the network. This issue could be handled by using unique names for the files (based on timestamps). Another solution would be to use a centralized authentication server, which makes a separate thread per requesting user.

In the next chapter we take a look at some scenarios in which we can use attribute-based authentication and we share our source code so others can take a look and improve upon the current implementation.

[2] More info on: http://freeradius.org/

CHAPTER 5

APPLICATIONS OF ATTRIBUTE-BASED WIFI AUTHENTICATION

In this chapter we present some real-world scenarios in which we can anonymously authenticate a WiFi connection using attribute-based authentication. We start by defining two classes of applications and continue by mentioning some possible example scenarios for each class. We end this chapter with a reference to our open-source implementation.

5.1 Classes of scenarios

We discuss two classes of scenarios for which attribute-based WiFi authentication can be used:

- One-time anonymous WiFi access: a user is allowed WiFi access only once.
- Subscription-based anonymous WiFi access: a user is allowed to the network as long as a certain subscription is valid.

The main difference between the two classes is the duration of such an access-granting credential. A one-time anonymous WiFi access credential invalidates after it has been used once or after a specific time period. A subscription-based WiFi access credential on the other hand is valid for multiple uses and it depends on the type of subscription when it stops granting the user access. A user only has this credential if he/she has a subscription for a particular service. For example a user has possession of a gym credential if he/she has a subscription for the gym. This credential is only valid until the user has a subscription, after that the credential invalidates.

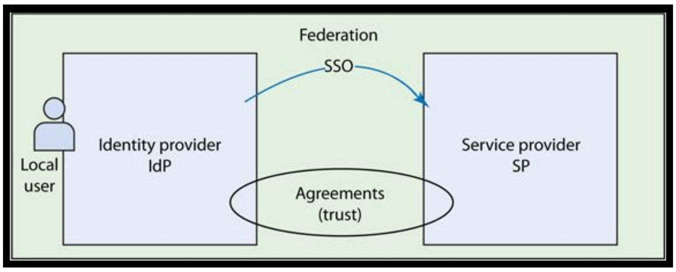

Figure 5.1 Federated access overview [45].

To clarify the concept of federated access consider a trusted party that knows the identities of some users. For example, a university knows the identities of all its students and academic staff. Companies that offer services for university members could use the institution as a trusted identity provider. The university handles authentication for these service providers. This means everyone who is in the university's records is allowed to use the service. A user requesting a service authenticates to the university and is then allowed to use the service if the authentication succeeds. The university is the identity provider, shares the identity information with the service providers and together form a federation as depicted in figure 5.1.

For another example of federated authentication, consider 'social login'. It is possible to authenticate (or sign in) at several online services, such as Expedia, by logging in via Google or Facebook. In these cases the social networks fulfill the role of identity providers.

In our case we do not have an identity provider but rather an attribute provider, or issuer. So a university issues a credential and this could be used at several service providers to authenticate and then use the requested service.

5.2 One-time anonymous WiFi access

In this part we look at two scenarios for one-time anonymous WiFi access.

5.2.1 WiFi in a flight

With the increase of mobile devices there are now also WiFi hotspots in newly built airplanes and it is expected that they will be added in current airplanes. As mentioned earlier, it is important to let users authenticate before allowing them on the WiFi network. Authentication should be anonymous and it should also be user friendly. Sending users a username/password combination via email prior to the flight is rather suboptimal, since:

- Users might not have internet access when they have boarded the plane;
- They might lose the email;
- It is not anonymous;
- It is prone to typing errors when the username or password is too hard/too long;

- Airlines need to maintain big databases of username/password combinations for each flight.

The solution we propose for is give the users (ie. passengers) their credentials on board of the airplane. This happens during boarding. When a passenger enters a plane he holds his phone to a credential issuing device and it generates a credential. Issuing a small credential (1 or 2 attributes) takes up to approximately 2.5 seconds on a smartcard [49]. On smartphones this is even faster. Credential issuance in this case is done using NFC, so the credential issuing device and the smartphone communicate using NFC and perform the issuing protocol from chapter 2. Issuance on laptops is a little tricky, since most laptops do not have an NFC chip. We could use Bluetooth instead or maybe via a dedicated and trusted USB device.

The issued credential would consist of a small number of attributes. It might contain:

- Type: for example, are you a passenger/crew member?
- Flight number
- Seat number or only the class in which the passenger is traveling (first, business or economy).

Having such a credential the user can connect to the WiFi network in the aircraft. The user associates with the access point and selectively discloses some attributes, the first and last would be sufficient in this case. Based on the type attribute some websites might be blocked and based on the class the connection speed may be faster. After authentication the user has internet access which he/she may use. The current implementation supports this scenario, it works good for smaller aircrafts where one access point suffices.

5.3 WiFi in supermarkets

Some supermarkets have their own mobile applications which customers may use to scan their products or to map the most efficient route through the store. These apps require the user to be in the supermarket and an effective method to verify whether this is the case, is to check if the user is connected to the in-store WiFi network. At the time of writing most WiFi networks in supermarkets are without authentication, so-called open-system WiFi networks. Using a network without authentication is bad practice and it would be better to restrict access to the network by using attribute-based credentials.

This scenario would be fairly similar to the first scenario. Customers that enter a supermarket obtain an is customer attribute via a similar NFC issuing device as was used while boarding. We assume that the number of customers who use their laptops in a supermarket is negligble, otherwise dedicated USB devices or Bluetooth would be a good method for them to acquire a credential. Association and authentication is also using the selective disclosure method. There are probably no different classes of users, so only that one attribute is sufficient and is part of the received credential.

Even though these applications have a strong resemblance at first sight, in practice there is a big difference between the two scenarios. In an airplane passengers usually do not move a lot, mobility is not a big issue. In a supermarket customers walk around a lot and they might get out of range of access point 1 and should almost instantaneous be connected to access point 2. In both applications issuance can be handled by NFC, this could automate the authentication process. A user who taps his phone to an issuance device gets the credential and a few seconds later is connected to the network, without the user actually setting up the connection. There are many other scenarios in this class for which attribute-based authentication could be used.

5.4 Federated/subscription-based anonymous WiFi access

In this section we look at two scenarios for federated or subscription-based anonymous WiFi access.

5.4.1 Eduroam

Eduroam is a worldwide service that allows students, researchers and academic staff to have WiFi access at their own and other institutions. It uses the Extensible Authentication Protocol and supports several types: EAP-TTLS or PEAP, together with MSCHAPv2 which requires a username/password. Alternatively, a user may choose for mutual authentication using X.509 certificates in EAP-TLS. Eduroam uses identity providers that hold data regarding what identities are allowed on the network. For example, the Radboud University has information regarding their students, they are allowed to use eduroam. It is possible for a student from a different university, TU Eindhoven for example, to use the eduroam network at the Radboud University. The RU's network contacts the identity provider from the TU and receives whether the user is allowed to use eduroam. Here we see the use of the federated access mechanism.

As mentioned earlier, there are disadvantages at authenticating with username and password. Authenticating using certificates is hard for individuals who are not technology savvy. Furthermore, they contain the username which is still identifying. A more privacy-friendly approach would be for the identity providers to issue credentials to the members of the institutions. These credentials could then be used for them to authenticate with the eduroam network.

This would also work at other national universities, since all the credentials are issued by the same (national) issuer. For international universities it is a little more difficult, since each country (or university) may have their own issuer. The visited institution is not able to directly check whether the credential was valid. In that case the university's eduroam network should 'contact' a party which is able to verify whether a user is allowed access to the network. In the credential there is an issuer identity stated, so it is easily possible for the visited university to know who to contact. This is how federated authentication at this moment works in Eduroam and this functionality might be inherited in its attribute-based variant. It is illustrated in figure 5.2.

5.4.2 Hotspots in cities

Larger cities have a great number of WiFi hotspots hosted by a lot of different parties. It is often referred to as the 'WiFi jungle'. Now internet service providers (ISPs), besides

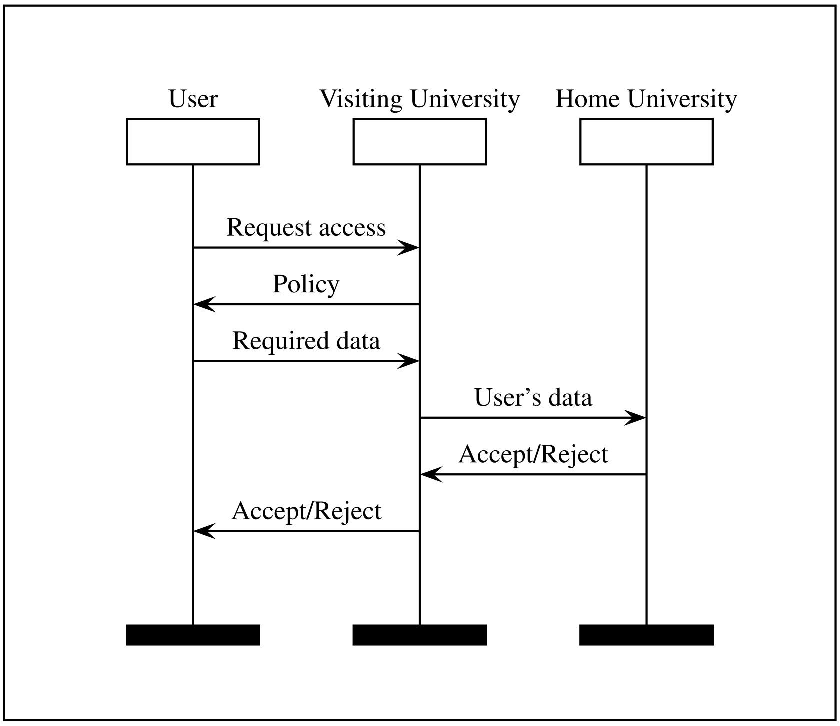

Figure 5.2 Federated attribute-based authentication for Eduroam.

providing internet access at home, also offer free WiFi hotspots for their customers, increasing the number of hotspots available. Let us consider Fon[1] as an example for a global WiFi hotspot network. It is used by KPN in the Netherlands to offer its customers free WiFi outside if they have a subscription at KPN. They can use the hotspots in large cities, or they may use access points located at other customer's homes.

Users of KPN Fon (and Fon in general) are able to download an application for their smartphone. If they have a valid subscription (either to KPN or to Fon directly) they are able to automatically connect to the free WiFi hotspots offered globally. However, the Fon and KPN applications are both coupled to a unique username, making users identifiable. The Fon mobile app is linked with the user's Google Play account (for the subscription) or iTunes account.

A privacy-preserving approach would be for KPN to issue the subscribed users a credential stating that they are allowed to use the Fon hotspots. The credential might contain the following attributes:

- Attribute stating that you are a KPN customer.
- Class of subscription: maybe if you pay a higher monthly fee, you would receive a faster connection.

[1] More information on: https://fon.com/

- Expiry date.

During authentication the user only sends the first and second attribute to the access points. Since it is based on a monthly subscription fee which the user needs to pay, it is possible to limit the validity of the credentials to 30 days. Just before the 30 days have elapsed, Fon can check whether another payment has been received. These payments are identifying, therefore it is possible to know what subscribers have paid and are allowed to renew their credential. If customer X has paid this is marked in KPN's systems. At that moment, customer X (or his/her device) can request a credential renewal from KPN using an identifier (either from the payment or from the customer). KPN can check whether X indeed made a payment, if that is the case KPN and X (or X's device) issue another credential together. Note, even though the customer sends an identifier, making the renewal process not anonymous, KPN does not know what credential was issued to X. Therefore, authentication remains anonymous.

5.5 Open-source implementation

In the above section we have seen only a small set of the scenarios possible with attribute-based WiFi authentication. Attribute-based WiFi authentication can replace current authentication schemes in almost every context and scenario, resulting in a more privacy-friendly authentication environment for the users.

Since we would like to invite others to incorporate attribute-based WiFi authentication in their setups, the code is open source available on https://www.aawsome.xyz. On this website there are also some examples explaining how to use this authentication scheme in some scenarios. As was mentioned earlier there are some improvements to be made to the current implementation. You are encouraged to download, use, improve the code and then submit it to replace the current version.

CHAPTER 6

CONCLUSIONS

In this chapter we summarize the key concepts of this thesis. We put the thesis in a broader context. In addition, we discuss possible drawbacks of using attribute-based WiFi authentication. We finish by stating some ideas for future research which can be conducted.

6.1 Summary

Authentication is important for WiFi hotspots, otherwise it would be possible for an adversary to eavesdrop on the traffic flowing through the network. In addition, it is possible to inject his own data into the network and possibly impersonating another user. Furthermore, these hotspots can be used to send custom commands to the clients connected to the access point. These commands can be used to order the clients to DDoS attack a particular system.

At the time of writing most public WiFi hotspots use WPA-2 enterprise together with the extensible authentication protocol (EAP). EAP provides a framework within which several authentication schemes can be used for the authentication process. In the authentication process three parties are involved:

- Supplicant: party requesting access to the network.
- Authenticator: party who gives the supplicant access to the network.

- Authentication server: party who decides what parties are allowed access to the network. The authentication server could be placed in the authenticator.

Most EAP authentication protocols require a username/password combination in order to authenticate with an access point. This makes the user and his/her activities traceable for the authenticator and authentication server. This is also the case when using a ticketing service, like Kerberos, for authenticating users.

In this thesis we presented a more privacy-friendly approach by authenticating using attribute-based credentials. A credential is a cryptographic container storing the user's attributes. Having such a credential, the supplicant (or user) may be allowed to a WiFi network. The user associates with the network it wishes to connect to. The authenticator sends a policy to the user. The policy states what attributes are required for accessing the network. The supplicant selectively discloses those attributes from the credential as requested by the authenticator. In addition, the supplicant does send a zero-knowledge proof of knowledge of the undisclosed attributes from the credential. The authenticator or authentication server only learn that the supplicant has a valid credential by receiving this proof and nothing more.

By authenticating using attribute-based credentials there are several benefits for the user:

- The user has control over the attributes, which are not stored in a centralized fashion. Therefore, it is not possible for a party to impersonate a user.
- The user does not disclose more information than required by the host.
- Since attributes are not necessarily identifying, it is not possible for any centralized party to collect traffic information regarding a user.
- The user does not have to memorize username/password combinations.

6.2 Discussion

In this section we discuss some drawbacks of authenticating using attribute-based credentials for public WiFi hotspots.

Networks that use attribute-based authentication require more advanced parties compared to networks using username and password authentication. Attribute-based authentication usess the issuance and selective disclosure protocols, as was discussed in chapter 2. These protocols involve big integer arithmetic, modular computation and are based on advanced cryptographic assumptions. It requires computationally strong parties which are able to store large parameters and compute with big integers within acceptable time. Otherwise it takes too long before a user is authenticated or has received his/her credential. This would prevent widespread usage due to usability issues. A network using username and password does not have to computationally strong. Usernames and password can be generated at random, they necessarily do now follow strict protocols.

Another downside with respect to the user's anonymity is the untraceability of the user's actions. Because a user is anonymous and multiple users may have authenticated using the

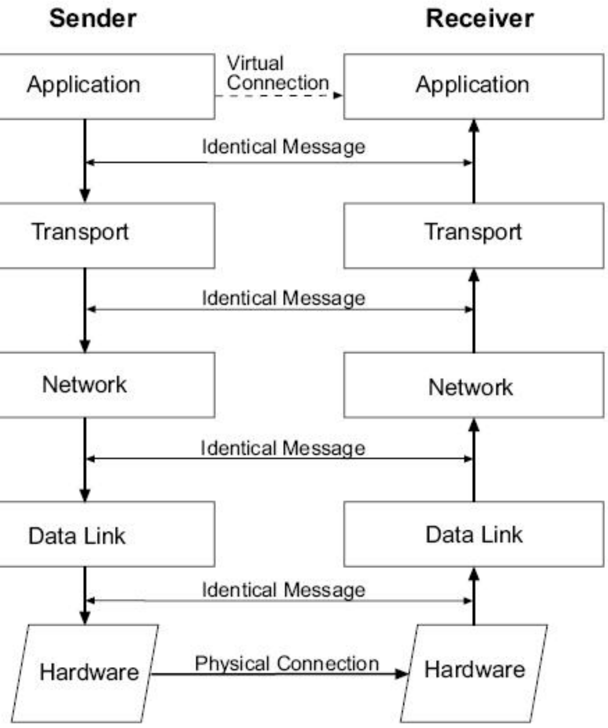

Figure 6.1 Protocol stack [4]

same attribute value, it is hard to trace particular actions back to a specific user. This is easier if every user has an attached identifier. This approach is however not privacy friendly. A solution for this issue is to move traceability and identification higher up the stack. Rather than having traceability and identification on the link/IP layer it would be better to have this on an application level.

Before we can discuss another drawback called MAC address traceability it is good to briefly look at how the internet works. We use the protocol stack as is illustrated in figure 6.1.

Consider an internet browser requesting a website. It sends an HTTP-GET request to the internet, in order for this message to arrive some work needs to be done. A webbrowser runs on the application layer and sends this message to the transport layer. The transport layer is a direct connection between two **processes** on different hosts. Processes run on ports which are active on the host, they are also identified by the ports on which they run. For example the webbrowser uses port X and requests website Y should get the data from Y's webserver back to port X in order for it to show the site to the user. The transport layer encapsulates the application level message (HTTP-GET) in a transport layer packet and gives it to the next layer.

The network layer gets the message from the transport layer and encapsulates it in a network layer packet. The network layer is used for **hosts** communicating with each other. Hosts are identified by their IP-addresses, the IP-address of the recipient is denoted in the Destination IP Address field of the packet. The two hosts in this example are: the computer on which the webbrowser runs and one of the servers on which the website runs.

The link layer sends datagrams (link layer packets) from one node to another. Between two hosts there might be several nodes in between (eg. access points, switches). The link layer is implemented in the network interfaces (eg. WiFi cards, ethernet adapters) of the hosts. Interfaces are referenced by their link layer addresses, so called MAC addresses. In contrast to IP addresses, MAC addresses are fixed and not build hierarchically. The link layer should translate the IP to MAC addresses, since it does not know about IP. The Address Resolution Protocol (ARP) is responsible for mapping IP addresses to their corresponding link layer equivalents. It asks Who has IP X? Tell IP Y and the interface who has that IP address responds with its MAC address. After receiving the right MAC address, this datagram is finally send as bits over a physical connection: by cabel or wireless. When the receiving party gets these bits, it sends everything up the protocol stack, decapsulating packets and sending it to higher levels. Eventually the webbrowser receives the webpage and shows it to the user.

So even though authentication is privacy-friendly using attribute-based credentials it is still possible for an adversary to collect data and link it to a specific user by knowing the MAC address from a different session. This also violates the user's privacy and defeats the purpose of authenticating in a privacy-friendly way. MAC addresses are part of the 802.11 standard so you cannot easily change the packets such that we do not have MAC addresses. There are however possibilities to cope with this problem. A user could prior to associating with a access point randomize his/her MAC address. Even though a MAC address is associated with a network interface, there is software available to change this address. That would lead to a per-session MAC address and results in a more difficult setting for an adversary to track the user cross sessions. Another more advanced solution is to encrypt message headers [30].

Another challenge associated with attribute-based credentials is revocation. Consider a user who is in possession of a credential stating that he has a subscription. If the user stops making payments or stops his subscription then that credential should be invalid. However, in contrast to a centralized authentication structure it is not possible for an issuer/verifier to drop that credential from a database. The credentials are only stored on the user's device. This means that a user may still authenticate using an invalid credential. This drawback can be addressed by having an expiry date with each credential. This date should however not be too far in the future because then renewal and checking whether a user still is a valid subscriber is not checked on a regular basis. There is ongoing research in this field of credential revocation [32].

6.3 Future work

In this section we mention some improvements for our implementation and some ideas for future research.

6.3.1 Improvements on implementation

At the moment of writing the implementation works only for a single credential. Authenticating should however function with an arbitrary number of credentials. This allows the verifier to construct more complex policies. In addition, authentication should not happen on the access point but rather on an authentication server. This would enable support for handoffs since the authentication server knows what MAC addresses have already been authenticated. Moreover, there should be support for multi-device authentication, if one user has a credential issued on device 1, then that credential should also be available on device 2. A credential management system is therefore required to make this easy for the users.

Our implementations could be incorporated in operating systems running on the supplicant and on the access point, instead of running them as standalone applications. This greatly improves usability. Usability could also be improved by providing a graphical user interface for WPA Supplicant, with which it is easier to select what attributes a user wishes to disclose.

Future research should also provide new ways of issuing credential aside from NFC. These ways should focus on offline issuance, where the user does not have an internet connection. We mentioned Bluetooth or a dedicated USB device which could be used. Research should show whether this is possible and what method is preferred in practice.

6.3.2 Attribute-based credentials

Delegation of attributes is currently computationally expensive, future research should show whether this could be done in a more efficient way. This would solve the problem mentioned in chapter 4 with relation to the use case in which a WiFi-access credential may be used by multiple people. There should also be a comparison research between the direct anonymous attestation and attribute-based WiFi authentication. These two authentication mechanisms are both implemented to use in the context of WiFi networks. They both provide more privacy for the user, so it is good to see what implementation is more efficient on smartphones/laptops and what provides better flexibility for various use cases.

Conducting research and finding improvements allow attribute-based WiFi authentication to be more robust and used in more environments. This increases the user's privacy in more WiFi networks and eventually results in untraceable WiFi access, which is awesome!

REFERENCES

1. hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator. https://w1.fi/hostapd/. Accessed 28th April 2016.
2. Identity mixer: What it does. http://www.research.ibm.com/labs/zurich/idemix/whatitdoes.html. Accessed 24th March 2016.
3. Linux wpa/wpa2/ieee 802.1x supplicant. https://w1.fi/wpa_supplicant/. Accessed 28th April 2016.
4. Network basics: Network layers. http://www.snmptools.net/netbasics/layers/. Accessed 20th May 2016.
5. IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, March 2012.
6. From 2016 to 5g. Technical report, Wireless Broadband Alliance, 2015.
7. Gergely Alpár. *Attribute-Based Identity Management: Bridging the Cryptographic Design of ABCs with the Real World.* PhD thesis, Radboud University Nijmegen, 2015.
8. Mark A Armstrong. Lagranges theorem. In *Groups and Symmetry*, pages 57–60. Springer, 1988.
9. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology-CRYPTO 2009*, pages 108–125. Springer, 2009.
10. David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios. In *Advances in Cryptology–ASIACRYPT 2012*, pages 626–643. Springer, 2012.

11. Felix Bieker, Marit Hansen, and Harald Zwingelberg. Towards a privacy-preserving inspection process for authentication solutions with conditional identification. *Proceeings of Open Identity Summit*, pages 85–96, 2014.

12. Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.

13. Mohamed Bourimi, Marcel Heupel, Dogan Kesdogan, and Thomas Fielenbach. Enhancing usability of privacy-respecting authentication and authorization in mobile social settings by using idemix (eu fp7 digital. me project technical report). 2011.

14. Stefan Brands. Untraceable off-line cash in wallet with observers. In *Advances in Cryptology-CRYPTO93*, pages 302–318. Springer, 1993.

15. Halil Ibrahim Bulbul, Ihsan Batmaz, and Mesut Ozel. Wireless network security: comparison of wep (wired equivalent privacy) mechanism, wpa (wi-fi protected access) and rsn (robust security network) security protocols. In *Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop*, page 9. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

16. Jan Camenisch. Direct anonymous attestation explained. Technical report, tech. rep., IBM Research, 2007.

17. Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 345–356, New York, NY, USA, 2008. ACM.

18. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *Security in communication networks*, pages 268–289. Springer, 2002.

19. Jyh-Cheng Chen, Ming-Chia Jiang, and Yi-wen Liu. Wireless LAN security and IEEE 802.11 i. *Wireless Communications, IEEE*, 12(1):27–36, 2005.

20. Jyh-Cheng Chen and Yu-Ping Wang. Extensible authentication protocol (EAP) and IEEE 802.1x: tutorial and empirical experience. *IEEE Communications Magazine*, 43(12):supl.26–supl.32, Dec 2005.

21. Liqun Chen, Dan Page, and Nigel P Smart. On the design and implementation of an efficient daa scheme. In *Smart Card Research and Advanced Application*, pages 223–237. Springer, 2010.

22. Ningning Cheng, Xinlei Wang, Wei Cheng, Prasant Mohapatra, and Aruna Seneviratne. Characterizing privacy leakage of public wifi networks for users on travel. In *INFOCOM, 2013 Proceedings IEEE*, pages 2769–2777. IEEE, 2013.

23. Credentials. Credentials idemix. https://github.com/credentials/credentials_idemix. Accessed 28th April 2016.

24. Ivan Damgård and Jesper Buus Nielsen. *Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor*. Springer, 2002.

25. Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In *Advances in CryptologyEUROCRYPT 2001*, pages 40–59. Springer, 2001.

26. Alpár et al. New Directions in IoT Privacy Using Attribute-Based AuthenticationPosition Paper. In *Proceedings of the 1st ACM Malicious Software and Hardware in Internet of Things Workshop (MAL-IoT16)*, 2016.

27. Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988.

28. Marc Fischlin. The cramer-shoup strong-rsa signature scheme revisited. In *Public Key CryptographyPKC 2003*, pages 116–129. Springer, 2003.

29. Keith Frikken, Mikhail Atallah, and Jiangtao Li. Attribute-based access control with hidden policies and hidden credentials. *Computers, IEEE Transactions on*, 55(10):1259–1270, 2006.

30. Ben Greenstein, Damon McCoy, Jeffrey Pang, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 40–53. ACM, 2008.

31. Linke Guo, Chi Zhang, Jinyuan Sun, and Yuguang Fang. Paas: A privacy-preserving attribute-based authentication system for ehealth networks. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 224–233. IEEE, 2012.

32. Jan Hajny and Lukas Malina. *Unlinkable attribute-based credentials with practical revocation on smart-cards*. Springer, 2012.

33. Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *Advances in CryptologyCRYPTO96*, pages 201–215. Springer, 1996.

34. Hyunuk Hwang, Gyeok Jung, Kiwook Sohn, and Sangseo Park. A study on mitm (man in the middle) vulnerability in wireless network using 802.1 x and eap. In *Information Science and Security, 2008. ICISS. International Conference on*, pages 164–170. IEEE, 2008.

35. Matthew Knysz, Xin Hu, Yuanyuan Zeng, and Kang G Shin. Open wifi networks: Lethal weapons for botnets? In *INFOCOM, 2012 Proceedings IEEE*, pages 2631–2635. IEEE, 2012.

36. Arash Habibi Lashkari, Mir Mohammad Seyed Danesh, and Behrang Samadi. A survey on wireless security protocols (wep, wpa and wpa2/802.11 i). In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 48–52. IEEE, 2009.

37. Hemanta K Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *Topics in Cryptology–CT-RSA 2011*, pages 376–392. Springer, 2011.

38. Kevin S McCurley. The discrete logarithm problem. In *Proc. of Symp. in Applied Math*, volume 42, pages 49–74, 1990.

39. Arunesh Mishra, Minho Shin, and William Arbaugh. An empirical analysis of the IEEE 802.11 MAC layer handoff process. *ACM SIGCOMM Computer Communication Review*, 33(2):93–102, 2003.

40. B Clifford Neuman and Theodore Ts' O. Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, 1994.

41. Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in CryptologyCRYPTO91*, pages 129–140. Springer, 1991.

42. F. Pereniguez, R. Marin-Lopez, G. Kambourakis, S. Gritzalis, and A.F. Gomez. Privakerb: A user privacy framework for kerberos. *Computers & Security*, 30(67):446 – 463, 2011.

43. Kai Rannenberg, Jan Camenisch, and Ahmad Sabouri. *Attribute-based Credentials for Trust: Identity in the Information Society*. Springer, 2014.

44. Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

45. Guruprasad Saralaya. Design and implement just-in-time provisioning with saml 2.0. https://www.ibm.com/developerworks/library/se-jitp/. Accessed 16th May 2016.

46. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Advances in cryptologyCRYPTO89 proceedings*, pages 239–252. Springer, 1989.

47. Daniel J Solove. A taxonomy of privacy. *University of Pennsylvania law review*, pages 477–564, 2006.

48. Adam Stubblefield, John Ioannidis, Aviel D Rubin, et al. Using the fluhrer, mantin, and shamir attack to break wep. In *NDSS*, 2002.

49. Pim Vullers and Gergely Alpár. Efficient selective disclosure on smart cards using idemix. In *Policies and Research in Identity Management*, pages 53–67. Springer, 2013.

50. Christian Wachsmann, Liqun Chen, Kurt Dietrich, Hans Löhr, Ahmad-Reza Sadeghi, and Johannes Winter. Lightweight anonymous authentication with tls and daa for embedded mobile devices. In *Information Security*, pages 84–98. Springer, 2010.

51. Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. In *Advances in Cryptology–EUROCRYPT 2005*, pages 19–35. Springer, 2005.

52. Kok-Kiong Yap, Yiannis Yiakoumis, Masayoshi Kobayashi, Sachin Katti, Guru Parulkar, and Nick McKeown. Separating authentication, access and accounting: A case study with openwifi. *Open Networking Foundation, Tech. Rep*, 2011.

53. Siyun Zhang and Jianwei Liu. A WLAN Anonymous Authentication Scheme Combining EAP-TLS and DAA. In *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*, pages 1232–1234. IEEE, 2012.