2021-06-08 theory

March 2021

1 Simultaneous Authentication of Equals

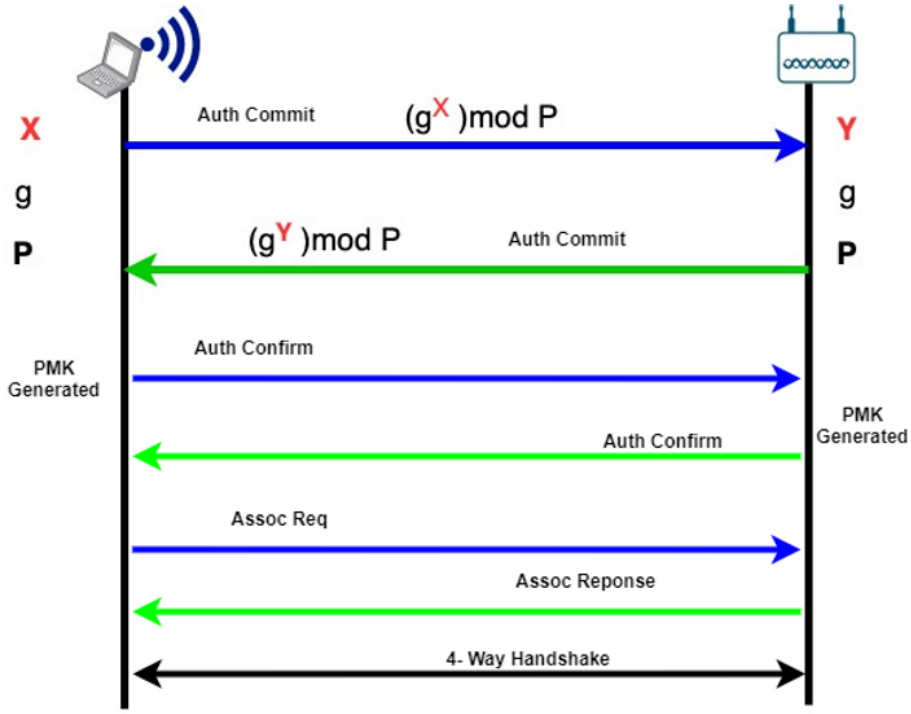
In cryptography, Simultaneous Authentication of Equals (SAE) is a secure password-based authentication and password-authenticated key agreement method. SAE is a variant of the Dragonfly Key Exchange defined in RFC 7664, based on Diffie–Hellman key exchange using finite cyclic groups which can be a primary cyclic group or an elliptic curve. The problem of using Diffie–Hellman key exchange is that it does not have an authentication mechanism. So the resulting key is influenced by a pre-shared key and the MAC addresses of both peers to solve the authentication problem.

The focus for SAE is to properly authenticate a device onto a network and using a password and MAC addresses to authenticate. An intruder should not be able to connect to a network unless they known password that the access point uses. Also, an intruder should not be able to guess either the password or the long-term authentication element of the password. In WPA-2 an intruder can listen to the 4-way handshake and can then brute force the password from hashed value created. With SAE — and which is based on a zero-knowledge proof known as dragonfly — we use the Diffie-Hellman key exchange method but adds an authentication element. With this we create a shared key with elliptic curve methods (NIST curves), and then use a pre-shared key and MAC addresses. When the client connects to the access point, they perform an SAE exchange. If successful, they will each create a cryptographically strong key, of which the session key will be derived from. If one session key is cracked it will only affect one key, and not all of the key used, as with WPA-2.

1.1 Step by step

In the commit phase, Alice (the client) generates two random values (a,A), then computes a scalar value (a+A). The value that Alice will pass is the PE (Password Equivalent — such as hashed value of the password that Bob and Alice know) raised to the power of -A. The operations are done with (mod q) and where q is a large prime number. Bob does the same, and then they exchange values. In the end they will have the same shared commit key (PE^{ab}). The password has been used to validate Bob and Alice, and they will only have the

shared shared commit value if they both have the same password. The intruder cannot determine either the original password or the final shared value. In the



confirm phase we then use the long-term key to generate a unique session key. An attacker cannot determine the password used or the long-term master key (PMK). The cracking of one session key will not reveal the rest of the keys which have been used. Which is not the case for WPA-2.

1.1.1 Algorithm

The SAE handshake supports both Finite Field Cryptography (FFC) using multiplicative groups modulo a prime (MODP groups), and it supports Elliptic Curve Cryptography (ECC) using Elliptic curve groups modulo a prime (ECP groups).(Algorithm are shown below Figure: 3). The WPA3 standard mandates

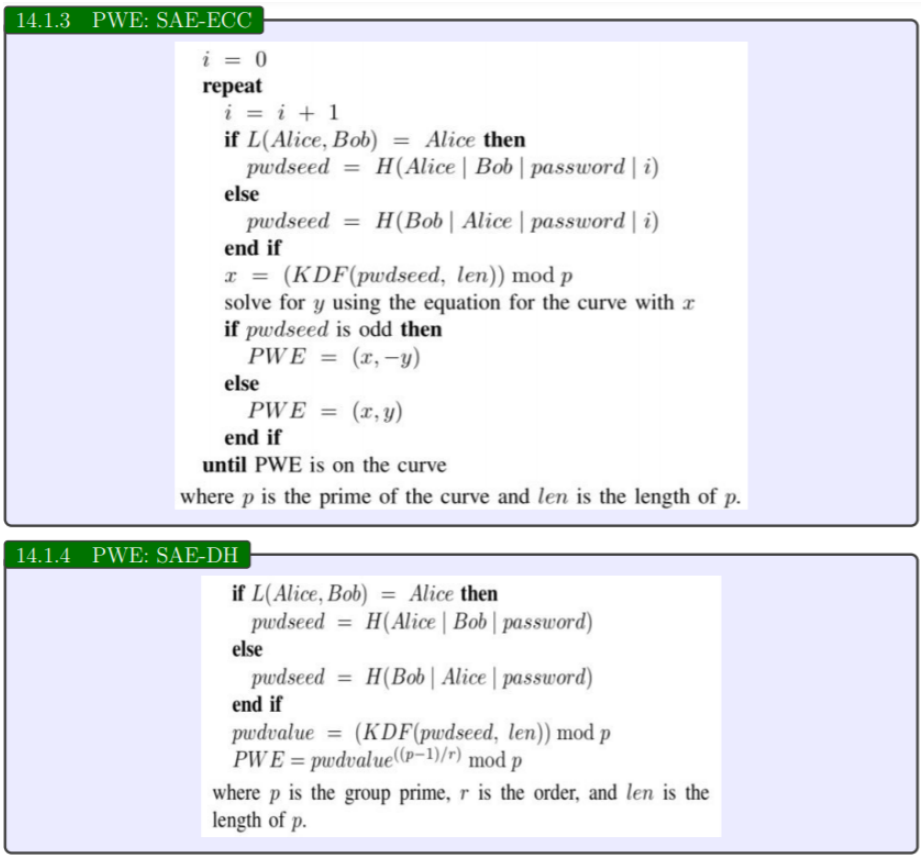


Figure 1:

the use of ECC for all Encryptions and all others are optional. This algorithm initialize a counter i to zero, then using the function L it determine which one is "greater" among ALice and Bob. Then the algorithm compute the hash value of Bob/Alice or Alice/Bob plus the password and the counter i. Through the function KDF, the hash value is stretched and it become the X coordinate to solve the equation for the curve. If the X value cannot resolve the equation, the loop is done again adding 1 to the i counter. After few iteration we should be able to get the value for the PWE.

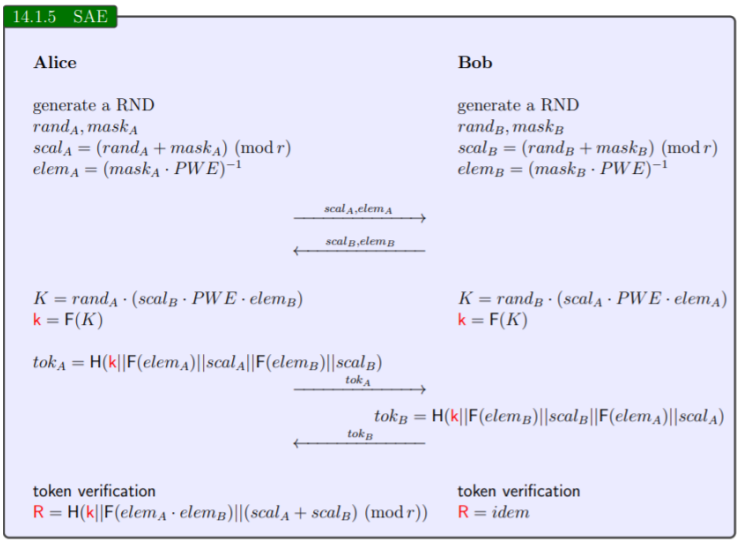


Figure 2: Here is rappresented the SAE-DH passage

The passage for the SAE-ECC differs in the construction of K, this is sown in the image below.

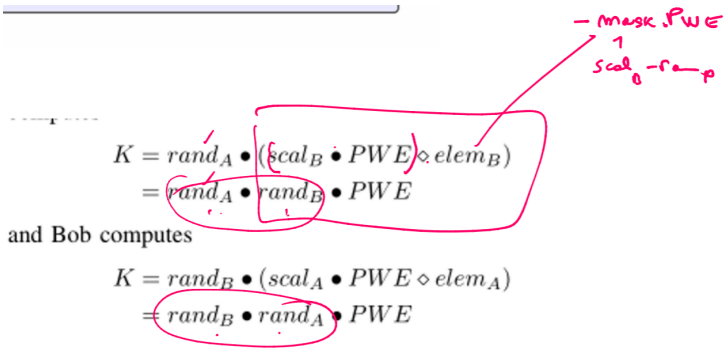


Figure 3: right K computation for ECC

2 Secure Remote Password

The Secure Remote Password protocol (SRP) is a cryptographically strong authentication protocol for password-based, mutual authentication over an insecure network connection. Authentication protocols are of two types:

- **plaintext-equivalent** = requires the server to store a copy of P or something from which P is computationally feasible to obtain.
- **verifier-based** = requires the server to store a V or something from which P is computationally infeasible to obtain. But from which P can be computationally verified.

SRP is verified-based which reduce the damage that a Trojan can inflict. The client, to be regarded as a human, as the P stored in his brain. The server store a verifier V that allows him to check P yet it is computationally infeasible to get P from the verifier V.

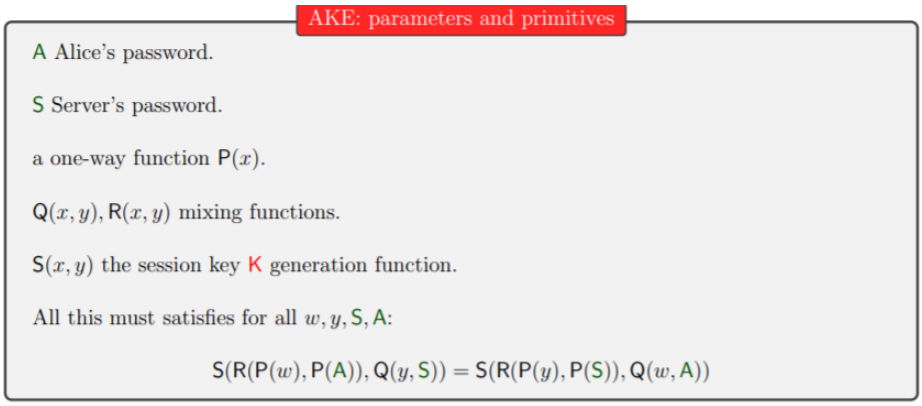


Figure 4:

To register, Alice sends her identity, a random *salt*, and a salted hash x of her password. Right from the start, you can see that a hash function is used (instead of a password hash function like Argon2) and thus anyone who sees this message can efficiently brute-force the hashed password. Not great. The use of the user-generated salt though, manage to prevent brute-force attacks that would impact all users.

The server can then register Alice by exponentiating a generator of a predetermined ring (an additive group with a multiplicative operation) with the hashed password. This is an important step as you will see that anyone with the knowledge of x can impersonate Alice.

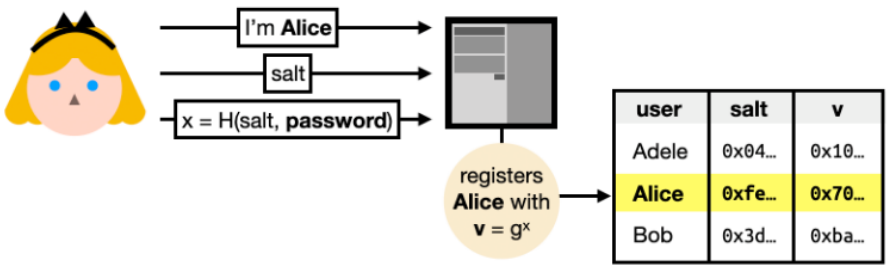


Figure 5:

What follows is the login protocol: You can now see why this is called

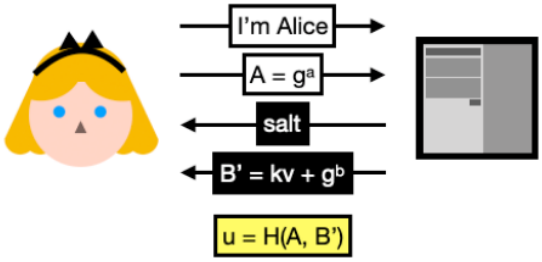


Figure 6:

a password authenticated key exchange, the login flow includes the standard ephemeral key exchange with a twist: the server's public key B' is blinded or hidden with v, a random value derived from Alice's password. (Note here k is a constant fixed by the protocol so we will just ignore it.)

Alice can only unblinds the server's ephemeral key by deriving v herself. To do this, she needs the *salt* she registered with (and this is why the server sends it back to Alice as part of the flow). For Alice, the SRP login flow goes like this:

- Alice re-computes x = H(salt, password) using her password and the salt received from the server.
- Alice unblinds the server's ephemeral key by doing B=$B' - kg^x = g^b$
- Alice then computes the shared secret **S** by multiplying the results of two key exchanges:
  1. B^a , the ephemeral key exchange
  2. $B^u x$, a key exchange between the server's public key and a value combining the hashed password and the two ephemeral public keys

Interestingly, the second key exchange makes sure that the hashed password and the transcript gets involved in the computation of the shared secret. But strangely, only the public keys and not the full transcript are used.

The server can then compute the shared secret **S** as well, using the multiplication of the same two key exchanges:

- A^b, the ephemeral key exchange
- $v^u b$, the other key exchange involving the hashed password and the two ephemeral public keys

The final step is for both sides to hash the shared secret and use it as the session key K = H(S). Key confirmation can then happen after both sides make successful use of this session key. (Without key confirmation, you're not sure if the other side managed to perform the PAKE.)