May 2022

Method to support iPSK for WPA3 clients as well as reduce Online Dictionary Attacks

Niranjan M M

Vijay Kothamasu

Nagaraj Kenchaiah

Method to support iPSK for WPA3 clients as well as reduce Online Dictionary Attacks

AUTHORS:
Niranjan M M
Vijay Kothamasu
Nagaraj Kenchaiah

ABSTRACT

WPA3 was developed with the backward compatibility into consideration, i.e., if WPA3 is enabled on WPA2+PSK SSID (also called mixed mode), then both WPA3 and WPA2-only clients can associate to same SSID. This will work as long as SSID is configured to use default-PSK. In other words, if SSID is configured with iPSK, then WPA2-only clients can associate to the SSID using iPSK, but WPA3 clients fails to associate to this SSID using iPSK, as current WPA3 SAE negotiation does not consider iPSK (unique PSK per client). Also, WPA3 was introduced to combat offline dictionary attacks on WPA2+PSK by using SAE protocol where-in an attacker would not be able to go through a word-list and compute a PMK that comes from the dragonfly handshake to test the MIC of a PTK off-line without interacting with the Authenticator. But still WPA3 is vulnerable to online dictionary attacks. The technique presented herein propose method to support iPSK even for the WPA3 clients and much more beneficial for the mixed-mode (i.e., supporting both WPA2 and WPA3 clients with iPSK) deployments. Also, this method decreases the attack surface of the WPA3 by aborting/breaking the SAE negotiation as early as possible.

DETAILED DESCRIPTION

As we know, WPA3 was developed with the backward compatibility into consideration, i.e., if WPA3 is enabled on WPA2+PSK SSID (also called mixed mode), then both WPA3 and WPA2-only clients can associate to same SSID. This will work as long as SSID is configured to use default-PSK. In other words, if SSID is configured with iPSK, then WPA2-only clients can associate to the SSID using iPSK, but WPA3 clients fails to associate to this SSID using iPSK, as current WPA3 SAE negotiation does not consider iPSK (unique PSK per client).

Also, WPA3 was introduced to combat offline dictionary attacks on WPA2+PSK by using SAE protocol where-in an attacker would not be able to go through a word-list and compute a PMK that comes from the dragonfly handshake to test the MIC of a PTK off-line without interacting with the Authenticator. But still WPA3 is vulnerable to online dictionary attacks.

The technique presented herein propose method to support iPSK even for the WPA3 clients and much more beneficial for the mixed-mode (i.e., supporting both WPA2 and WPA3 clients with iPSK) deployments. Also, this method decreases the attack surface of the WPA3 by aborting/breaking the SAE negotiation as early as possible. With this method, iPSK for WPA3 clients is supported as well as it reduces the 'online dictionary attacks' on WPA3 SAE protocol by introducing a communication mechanism (Access-Request/Access-Accept) with the Authentication Server to fetch iPSK for corresponding client's mac-address as well as using One Time Cookie (OTC), which is shared in encrypted form between Authenticator and Supplicant. Further, this OTC would be validated by Authenticator before sending "Authentication (SAE Confirm)" message back to Supplicant. If validation fails, Authenticator abort/break the SAE session and sends "SAE Confirm Failure" to the Supplicant.

As per this method, upon receiving "Authentication (SAE Commit") from the Supplicant (client), Authenticator (WLC) communicate with the Authentication Server (AAA server) to get iPSK corresponding to the mac-address of the client. Along with iPSK, Authentication server generates unique "One Time Cookie (OTC)" and send it to Authenticator. Authenticator uses iPSK to generate parameters (Password Element [PE], kck, mk etc.,) for SAE (Note: "mk" is used as PMK for 802.11i four-way handshake later). Since this method retrieves iPSK corresponding to the client mac-address from the Authentication Server, iPSK clients can associate to the WPA3 enabled SSID as well as to overcome the limitation of mixed mode (where WPA3 and WPA2 clients can be associated to SSID where-in both WPA3 and WPA2+PSK are configured. This also helps to reduce attack surface where attacker is trying with different mac-address and different pass-phrase combinations. Later, Authenticator encrypts the OTC using Password Element (PE) and sends to Supplicant as part of "Authentication (SAE Commit)" message. Upon receiving "SAE Commit", Supplicant generate parameters (kck, mk etc.,) for SAE and further decrypts the OTC using "Password Element". Further, Supplicant generate and send "Authentication (SAE Confirm)" message along with OTC encrypted using "mk" to the Authenticator. Upon receiving "SAE Confirm" message, Authenticator decrypt the OTC using "mk" and validates the OTC. If validation is successful, then it sends "Authentication (SAE Confirm)" message back to Supplicant. If OTC validation fails, it aborts/breaks the SAE session and sends "SAE Confirm Failure" to Supplicant. Since SAE session aborted upon OTC validation failure, it reduces the attack surface of WPA3 for Online Dictionary Attack.

Figure-1 describe method to support iPSK for WPA3 clients and also reduce Online Dictionary Attacks.

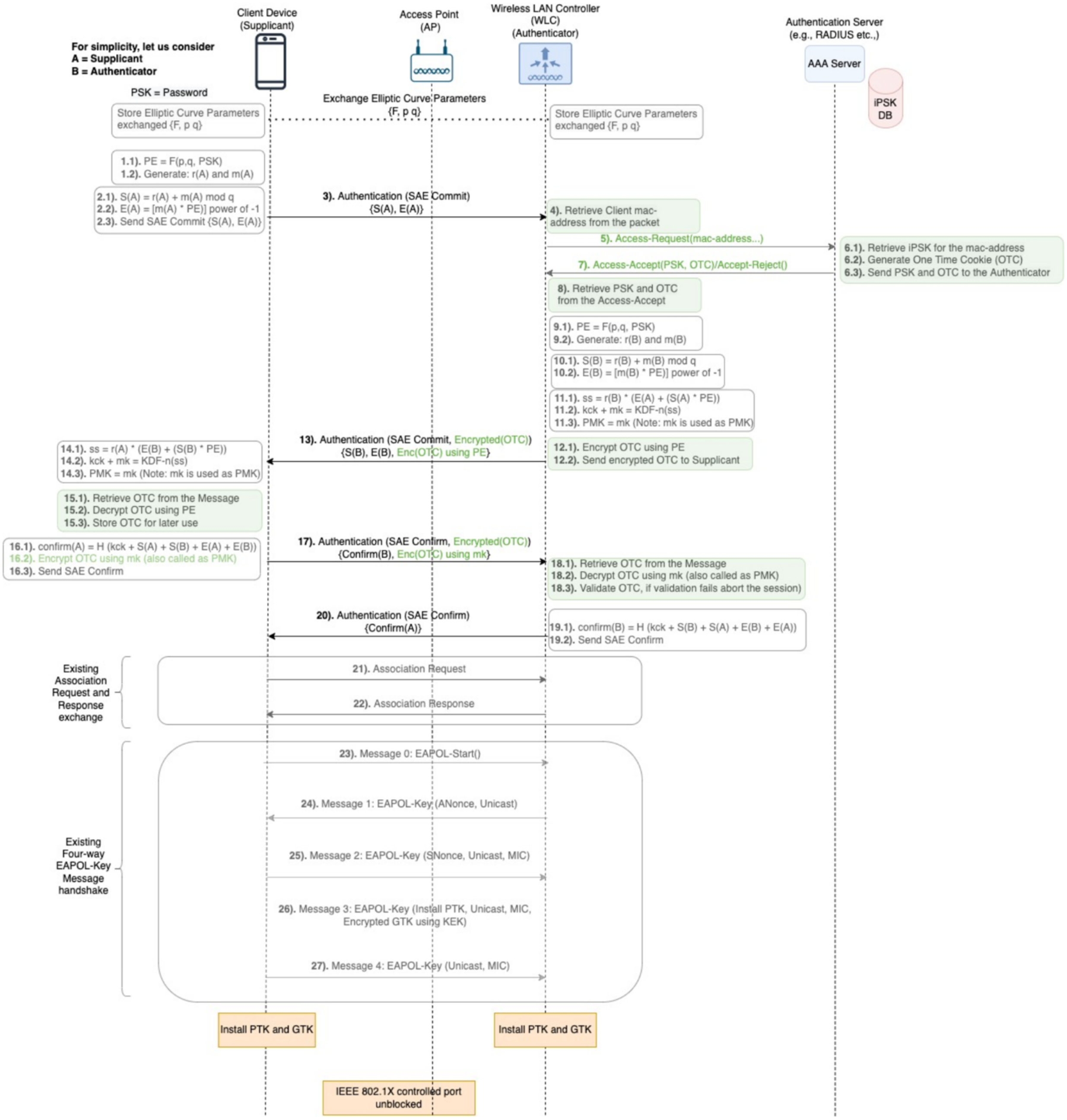
Client Device
(Supplicant)
Access Point
(AP)
Wireless LAN Controller
(WLC)
(Authenticator)
Authentication Server
(e.g., RADIUS etc.,)
AAA Server
iPSK
DB
For simplicity, let us consider
A = Supplicant
B = Authenticator
PSK = Password
Store Elliptic Curve Parameters exchanged {F, p q}
Exchange Elliptic Curve Parameters {F, p q}
Store Elliptic Curve Parameters exchanged {F, p q}
1.1). PE = F(p,q, PSK)
1.2). Generate: r(A) and m(A)
2.1). S(A) = r(A) + m(A) mod q
2.2). E(A) = [m(A) * PE)] power of -1
2.3). Send SAE Commit {S(A), E(A)}
3). Authentication (SAE Commit) {S(A), E(A)}
4). Retrieve Client mac-address from the packet
5). Access-Request(mac-address...)
6.1). Retrieve iPSK for the mac-address
6.2). Generate One Time Cookie (OTC)
6.3). Send PSK and OTC to the Authenticator
7). Access-Accept(PSK, OTC)/Accept-Reject()
8). Retrieve PSK and OTC from the Access-Accept
9.1). PE = F(p,q, PSK)
9.2). Generate: r(B) and m(B)
10.1). S(B) = r(B) + m(B) mod q
10.2). E(B) = [m(B) * PE)] power of -1
11.1). ss = r(B) * (E(A) + (S(A) * PE))
11.2). kck + mk = KDF-n(ss)
11.3). PMK = mk (Note: mk is used as PMK)
12.1). Encrypt OTC using PE
12.2). Send encrypted OTC to Supplicant
13). Authentication (SAE Commit, Encrypted(OTC)) {S(B), E(B), Enc(OTC) using PE}
14.1). ss = r(A) * (E(B) + (S(B) * PE))
14.2). kck + mk = KDF-n(ss)
14.3). PMK = mk (Note: mk is used as PMK)
15.1). Retrieve OTC from the Message
15.2). Decrypt OTC using PE
15.3). Store OTC for later use
16.1). confirm(A) = H (kck + S(A) + S(B) + E(A) + E(B))
16.2). Encrypt OTC using mk (also called as PMK)
16.3). Send SAE Confirm
17). Authentication (SAE Confirm, Encrypted(OTC)) {Confirm(B), Enc(OTC) using mk}
18.1). Retrieve OTC from the Message
18.2). Decrypt OTC using mk (also called as PMK)
18.3). Validate OTC, if validation fails abort the session)
19.1). confirm(B) = H (kck + S(B) + S(A) + E(B) + E(A))
19.2). Send SAE Confirm
20). Authentication (SAE Confirm) {Confirm(A)}
Existing Association Request and Response exchange
21). Association Request
22). Association Response
Existing Four-way EAPOL-Key Message handshake
23). Message 0: EAPOL-Start()
24). Message 1: EAPOL-Key (ANonce, Unicast)
25). Message 2: EAPOL-Key (SNonce, Unicast, MIC)
26). Message 3: EAPOL-Key (Install PTK, Unicast, MIC, Encrypted GTK using KEK)
27). Message 4: EAPOL-Key (Unicast, MIC)
Install PTK and GTK
Install PTK and GTK
IEEE 802.1X controlled port unblocked

Figure-1

The technique presented herein is explained in detail as below:

- As part of WPA3 SAE protocol, Supplicant and Authenticator exchange and store Elliptic Curve Parameters (p, q).
- Note: Since SAE protocol flow uses multiple independent parameters both on Supplicant and Authenticator, for simplicity, suffixing the parameters of Supplicant with "A" and the parameters of Authenticator with "B".
- On Supplicant (Client):
    - Supplicant generate Password Element (PE) using negotiated algorithm F and Elliptic Curve Parameters (p, q). It also generates r(A) and m(A).
    - Later r(A) and m(A) are used to generate S(A) and E(A) as mentioned in the above diagram.
    - Supplicant sends "Authentication (SAE Commit)" message along with S(A) and E(A) to the Authenticator.
- On Authenticator (WLC):
    - Upon receiving SAE Commit, Authenticator retrieves the S(A), E(A) and Client mac-address. Stores S(A) and E(A) for later use.
    - Authenticator sends Access-Request with Client mac-address to the Authentication Server (AAA server). [In some deployments, local AAA server would be running on the Authenticator itself].
- On Authentication Server (AAA server):
    - Upon receiving Access-Request, AAA server retrieve iPSK corresponding to the mac-address of the Client as well as generate unique One Time Cookie (OTC) having details of client authentication record as well as expiry time/date.
    - AAA server sends Access-Accept along with retrieved PSK and OTC to the Authenticator.
    - If iPSK to mac-address is not found, then AAA server sends Access-Reject to Authenticator.
- On Authenticator (WLC):
    - Upon receiving Access-Accept, Authenticator retrieves PSK and OTC.
    - Authenticator uses PSK to generate Password Element (PE) using earlier stored (p, q) using function F().
    - Also, Authenticator generates r(B) and m(B).

  - r(B) and m(B) are used for generating S(B) and E(B) as mentioned in the above diagram.
  - Further, S(A) and E(A) received earlier from the Supplicant are used to generate ss and kck as below:
    - ss = r(B) * (E(A) + (S(A) * PE))
    - kck + mk = KDF-n(ss).
  - Authenticator sends "Authentication (SAE Commit)" message along with S(B) and E(B) to the Supplicant.
  - This message also carries One Time Cookie (OTC) received from the AAA server, which is encrypted using Password Element (PE).
- On Supplicant (Client):
  - Upon receiving SAE Commit, Supplicant retrieves the S(B), E(B) and encrypted OTC.
  - Further, Supplicant uses S(B) and E(B) to generate ss and kck as below:
    - ss = r(A) * (E(B) + (S(B) * PE))
    - kck + mk = KDF-n(ss).
  - Also Supplicant decrypts OTC using Password Element (PE) and store OTC for later use.
  - Further, Supplicant generate and send "Authentication (SAE Confirm)" message with confirm(A) = H(kck + S(A) + S(B) + E(A) + E(B)) to the Authenticator.
  - This message also carries OTC encrypted using "mk".
- On Authenticator:
  - Upon receiving "SAE Confirm", Authenticator retrieves encrypted OTC from the message. Decrypt the OTC using "mk" generated earlier.
  - Authenticator validates the OTC. If validation is success, it will send "Authentication (SAE Confirm)" message with confirm(B) = H (kck + S(B) + S(A) + E(B) + E(A)) to the Supplicant. If validation fails, abort the session and send "SAE Failure" back to the Supplicant.
  - Since SAE session aborted upon OTC validation failure, it reduces the attack surface of WPA3 for Online Dictionary Attack.
- On Supplicant:
  - Upon receiving SAE Confirm, Supplicant would go for association phase and followed by EAPOL four-way handshake process as per the existing art.

In summary, the techniques presented herein propose method wherein, the One Time Cookie (OTC) shared between Authentication Server, Authenticator and Supplicant reduces the Online Dictionary Attack on WPA3. OTC is always sent in encrypted form between Authenticator and Supplicant, hence through OTC validation, validating the SAE messages exchanged. If validation failed, aborting/breaking the current SAE session. Moreover, no additional configuration needed for this mechanism and keep it simple, but still WPA3 clients can use iPSK while associating to WPA3 SSID and hence much more secure. Also, existing messages are leveraged to carry attestation information between Authenticator and Supplicant, which will keep the implementation simple. Additionally, with mixed-mode (i.e., SSID configured to support both WPA3 and WPA2+PSK), both types of clients can use iPSK (unique PSK per client) for connectivity.