

Weaknesses in real-world protocols

Martin Stanek

Department of Computer Science
Comenius University
`stanek@dcs.fmph.uniba.sk`

Cryptology 1 (2022/23)

Content

KRACK

Dragonfly (SAE)

Bluetooth

KRACK

- ▶ Key Reinstallation Attacks (Vanhoef, Piessens, 2017)
 - ▶ just an idea
 - ▶ details and paper available at www.krackattacks.com
- ▶ WPA (Wi-Fi Protected Access)
 - ▶ WPA – 802.11i (draft D3.0); WPA2 – 802.11i (final version D9.0)
 - ▶ two data confidentiality and integrity protocols: (WPA-)TKIP and (AES-)CCMP
 - ▶ 802.11ad amendment: Galios/Counter Mode Protocol (GCMP)
- ▶ 4-way handshake protocol
 - ▶ mutual authentication based on PMK (Pairwise Master Key)
 - ▶ PMK derived from preshared secret (WPA-Personal) or negotiated in 802.1x (WPA-Enterprise)
 - ▶ establish a session key PTK (Pairwise Transient Key)
- ▶ supplicant/station (client) and authenticator (AP)

4-way handshake

- ▶ simplified presentation
- ▶ 4-way handshake:
 1. $AP \rightarrow S$: ANonce (now the supplicant can derive PTK)
 2. $S \rightarrow AP$: SNonce, MIC_{KCK} (now the authenticator can derive PTK)
 3. $AP \rightarrow S$: GTK, MIC_{KCK} (GTK encrypted with KEK)
 4. $S \rightarrow AP$: Ack, MIC_{KCK} (Ack)
- ▶ MIC (Message Integrity Check)
- ▶ GTK (Group Temporal Key ... broadcast/multicast)
- ▶ PTK = $PRF(PMK, AP_{Mac}, S_{Mac}, ANonce, SNonce)$, divided into
 - ▶ KCK (EAPOL-Key Confirmation Key) – for MIC computation
 - ▶ KEK (EAPOL-Key Encryption Key) – for encryption of GTK
 - ▶ TK (Temporal Key) – for encryption of data frames
 - ▶ TMK1, TMK2 (Temporal AP MIC Key) – keys for MIC computation (unicast), one for each direction

KRACK – idea

- ▶ remark: offline dictionary attack (4th message), no forward secrecy
- ▶ the third (or the first) message can be retransmitted (multiple times)
 - ▶ for example, if the authenticator does not receive message 4 (or 2)
 - ▶ reinstall the PTK and reset initialization vector (nonce) for data encryption and authentication
 - ▶ according 802.11i “AP retransmits message 1 or 3 if it did not receive a reply”
- ▶ behavior of clients differs (depends on NIC and supplicant implementation)
- ▶ other variants: key reinstallation against group key handshake ...

KRACK – impact

- ▶ CCMP – AES-CCM (CTR and CBC-MAC)
 - ▶ key and IV are re-used, i.e. keystream is re-used
 - ▶ attacker can decrypt
- ▶ GCMP – AES-GCM
 - ▶ keystream re-use
 - ▶ authentication key can be recovered after nonce reuse (Joux, 2006)
 - ▶ attacker can decrypt and inject own data
- ▶ special weakness in Android and Linux:
 - ▶ retransmitted message 3 causes all-zero key
- ▶ other variants of KRACK attack (2018)

Dragonfly (SAE)

- ▶ WPA3 (2018)
- ▶ mandatory: new protocol Simultaneous Authentication of Equals (SAE)
- ▶ original design – Harkins (2008)
 - ▶ balanced PAKE protocol
 - ▶ IEEE 802.11-2016
 - ▶ RFC 7664 (Dragonfly Key Exchange)
 - ▶ other variants: EAP-pwd (RFC 5931), IKEv2 Secure PSK Authentication (RFC 6617)
- ▶ EAP-pwd: can be used in some enterprise WiFi networks
- ▶ SAE is used to derive new PMK for the 4-way handshake
 - ▶ does not prevent KRACK per-se
 - ▶ prevents offline dictionary attack
 - ▶ ensures forward secrecy
- ▶ M. Vanhoef, E. Ronen: *Dragonblood: Attacking the Dragonfly Handshake of WPA3* (2019) – weaknesses in SAE and EAP-pwd

Dragonfly (SAE) – introduction

- ▶ simplified for presentation
- ▶ main goals and properties
 - ▶ no fixed roles (e.g. initiator, client, server, ...)
 - ▶ both parties can initiate the protocol (simultaneously)
 - ▶ forward secrecy
 - ▶ resistance to offline dictionary attack (and other attacks)
 - ▶ based on DLOG problem
- ▶ proposed for modular and elliptic curves groups
 - ▶ parameters: primes p , q , and $q \mid (p - 1)$
 - ▶ modular group: subgroup of order q is used
 - ▶ elliptic curve group over $\text{GF}(p)$: group order q , curve $y^2 = x^3 + ax^2 + b \bmod p$
- ▶ H – hash function (random oracle); KDF – key derivation function

Dragonfly (SAE) – password element P

- ▶ map password pw to a group element P

- ▶ hash to group:

```
for counter in range(1, 256):  
    seed =  $H(addr_A, addr_B, pw, counter)$   
     $x = KDF(seed, p)$   
    if  $x \geq p$ : continue  
     $P = x^{(p-1)/q} \bmod p$   
    if  $P > 1$ : return  $P$ 
```

- ▶ hash to curve:

```
base =  $pw$ , counter = 1  
while counter++ < 40 or  $P$  not found:  
    seed =  $H(addr_A, addr_B, base, counter)$   
     $x = KDF(seed, p)$   
    if  $x \geq p$ : continue  
    if  $x^3 + ax + b \in QR_p$  and  $P$  not found:  
         $P = (x, \text{sqrt}(x^3 + ax + b) \bmod p)$   
        base = random()  
return  $P$ 
```

SAE – protocol

1. Commit Exchange (presentation uses elliptic curves)

- ▶ A select random $r_A, m_A \in \mathbb{Z}_q^*$;
 A computes $s_A = (r_A + m_A) \bmod q$, and $E_A = -m_A \cdot P$
- ▶ B select random $r_B, m_B \in \mathbb{Z}_q^*$;
 B computes $s_B = (r_B + m_B) \bmod q$, and $E_B = -m_B \cdot P$

$A \rightarrow B: s_A, E_A$

$B \rightarrow A: s_B, E_B$

- ▶ check validity of s_X , check that E_X is on the curve
- ▶ shared secret element K is computed:

$$A: K = r_A \cdot (s_B \cdot P + E_B) = r_A \cdot ((r_B + m_B) \cdot P - m_B \cdot P) = (r_A r_B) \cdot P$$

$$B: K = r_B \cdot (s_A \cdot P + E_A) = r_B \cdot ((r_A + m_A) \cdot P - m_A \cdot P) = (r_A r_B) \cdot P$$

- ▶ shared key $k = H(K)$

SAE – protocol (2)

2. Confirmation Exchange

- ▶ verify k and transcript of the protocol:

$$A \rightarrow B: \quad c_A = \text{HMAC}_k(s_A, E_A, s_B, E_B)$$

$$B \rightarrow A: \quad c_B = \text{HMAC}_k(s_B, E_B, s_A, E_A)$$

- ▶ variants of Dragonfly differ in
 - ▶ computation of password element
 - ▶ computation of confirmation messages
 - ▶ key derivation and usage (e.g. multiple keys are derived), ...

SAE – some earlier results

- ▶ D. Clarke, F. Hao: Cryptanalysis of the Dragonfly Key Exchange Protocol (2013)
 - ▶ offline dictionary attack for small subgroups
 - ▶ important to perform checks in “Commit Exchange” step (validity of E_X and s_X)
- ▶ J. Lancrenon, M. Škrobot: On the Provable Security of the Dragonfly Protocol (2015)
 - ▶ security proof in model by Bellare, Pointcheval and Rogaway (other models exist)
 - ▶ assumptions: random oracle model (H), CDH, DIDH (Decisional Inverted-Additive Diffie-Hellman)
 - ▶ DIDH: hard to distinguish $g^{1/(x+y)}$ and a random $g^{1/z}$ when given $g^{1/x}$ and $g^{1/y}$.

Timing attacks – MODP groups

- ▶ hash to group – number of iterations depends on password
 - ▶ KDF returns bit string of length $|p|$
 - ▶ probability that $x \geq p$ is not negligible for some groups
 - ▶ RFC 5114 – group 22 (30.84%), group 23 (32.40%), group 24 (47.01%)
 - ▶ Is the difference between r and $r + 1$ iterations measurable?
Yes (see the experiments in Dragonblood paper)
e.g. for group 22 ≈ 75 measurements were enough to identify r
 - ▶ number of iteration depends on MAC addresses as well
 - ▶ spoofing MAC, measuring iterations ... building a password “profile”
 - ▶ offline dictionary/brute-force attack

Timing attacks – elliptic curves

- ▶ hash to curve for EAP-pwd
 - ▶ iterate until P is on the curve
 - ▶ similar timing leak as for hash to group
- ▶ hash to curve for SAE – timing attacks countermeasures already present
 - ▶ $x \geq p$ is not negligible for Brainpool curves (RFC 6932)
 - ▶ multiple measurements for a MAC:
 - more iteration with real password yield lower variance
 - average time depends on real iterations and number of $x \geq p$ results (see the experiments in Dragonblood paper)
 - ▶ cache attacks (Flush and Reload)
 - ▶ blinding the y value in the QR test
 - ▶ detection of QR test result in the first iteration
 - ▶ assumption: attacker runs a process on victim host (e.g. Android app)

Other issues and observations

- ▶ AP must store the password in plaintext
- ▶ WPA3 Transition Mode – AP accepts WPA3-SAE and WPA2 with the same password
 - ▶ compatibility with old clients
 - ▶ downgrade attack are detected, thanks to properties of 4-way handshake
 - ▶ attack has enough data for offline dictionary attacks
 - ▶ countermeasure: remember if the network supports WPA3-SAE (“pinning”)
- ▶ high overhead of hash to curve
 - ▶ timing attacks defense (40 iterations) is costly for lightweight devices
 - ▶ existing DoS countermeasures can be defeated
e.g. experiment with 8 connections/s – AP’s CPU saturated
- ▶ fatal impact of bad PRNG
 - ▶ attacker reconstructs P and K
 - ▶ impact worse than bad PRNG in WPA2
- ▶ update to WPA3?

Bluetooth

- ▶ widely deployed protocol
 - ▶ mobile phones, laptops, fitness/smart watches, headphones, ...
- ▶ two protocols (similar):
 - ▶ Bluetooth BR/EDR – Secure Simple Pairing (SSP)
 - ▶ Bluetooth Low Energy – Low Energy Secure Connection (LE SC)
- ▶ goals for both protocols: confidentiality and MITM protection
- ▶ authenticated ECDH key exchange
- ▶ both protocols are vulnerable
- ▶ E.Biham, L. Neumann: *Breaking the Bluetooth Pairing – Fixed Coordinate Invalid Curve Attack* (2018)
- ▶ other attacks for older versions exist (e.g. crackle)

Invalid Curve Attack on ECDH

- ▶ ECDH (elliptic curve E , generator P):
 1. $A \rightarrow B: U = u \cdot P$
 2. $B \rightarrow A: V = v \cdot P$ \Rightarrow shared key: $K = (uv) \cdot P$
- ▶ attacker uses invalid points (not on the curve) to find shared key
 - ▶ group operation does not depend on b ($y^2 = x^3 + ax^2 + b$), see the “dlog” lecture
 - ▶ attacker can choose a curve E' (different b') with subgroup of small order
 - ▶ let P' be a generator, and q' is the order

Invalid Curve Attack on ECDH (2)

- ▶ attack:

1. $A \rightarrow M: U = u \cdot P$
2. $M \rightarrow A: P'$... A computes $K = u \cdot P'$
- ... $A \rightarrow M: c = E_K(m)$

- ▶ assumption: M knows m

- ▶ M finds $u' \in \mathbb{Z}_{q'}: E_{u' \cdot P'}(m) = c \Rightarrow u \equiv u' \pmod{q'}$

- ▶ recovering u :

- ▶ iterate attack for multiple times for different (co-prime) q'
- ▶ use CRT to compute u

- ▶ assumptions:

- ▶ the protocol can be executed multiple times and u does not change
- ▶ attacker can choose arbitrary P'

- ▶ Bluetooth specification: to prevent this attack, refresh your parameters for every pairing

Fixed Coordinate Invalid Curve Attack (idea)

- ▶ let's ignore all other SSP / LE SC details
- ▶ main problem:
y-coordinate is not authenticated (only x-coordinate of “public key”)
- ▶ semi-passive attack:
 - ▶ set y-coordinate of both public keys to 0 (a curve with different b')
 - ▶ the order of these points is 2
 - ▶ if both “private keys” are even (prob. 25%), then $K = 0$ (point at infinity)
 - ▶ attacker knows the shared key (shared by both parties)
- ▶ fully-active attack:
 - ▶ improved attack with 50% probability of success
- ▶ large majority of the Bluetooth devices are vulnerable (CVE-2018-5383)
 - ▶ chips/implementations: Broadcom, Qualcomm, Intel / Apple, Google, ...