Karlsruher Institut für Technologie

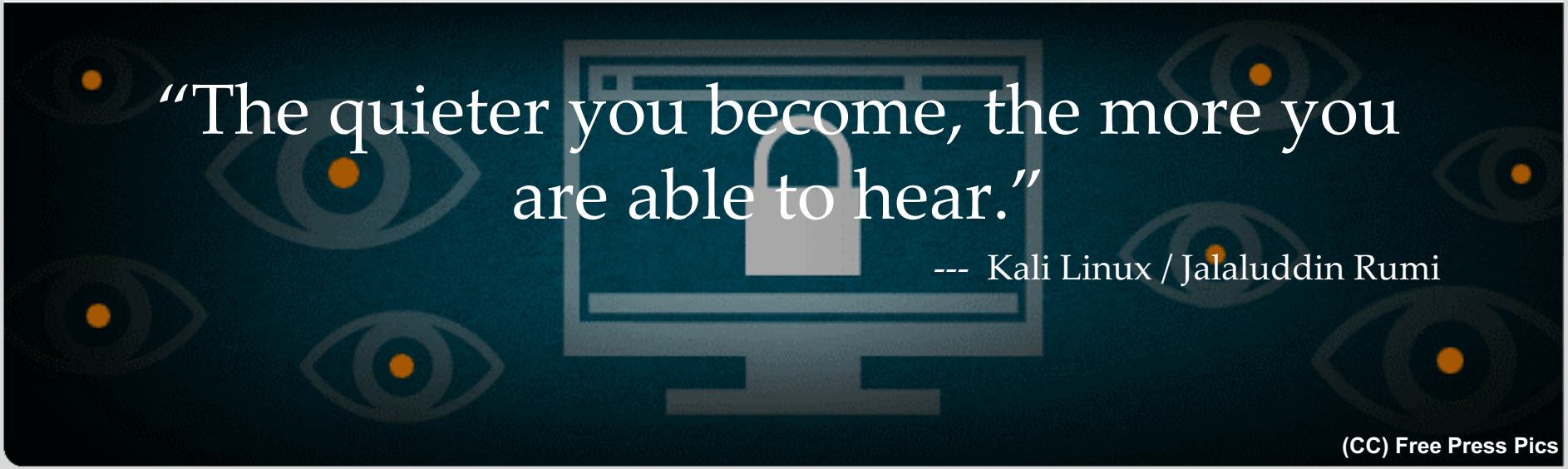
WPA2 Handshake and Half-Handshake Attacks

KASTEL-Praktikum Sicherheit

Benny Görzig (bgoerzig@gmail.com), Florian Loch (me@fdlo.ch)

Betreuer: Erik Krempel, Pascal Birnstill (Fraunhofer IOSB), Daniel Keller (LKA BW)

KOMPETENZZENTRUM FÜR ANGEWANDTE SICHERHEITSTECHNOLOGIE (KASTEL)



Agenda

1. Einleitung
2. WPA2-Cracking
3. Handshakes mitschneiden
4. Fazit & Ausblick

1. Einleitung

2. WPA2-Cracking
3. Handshakes mitschneiden
4. Fazit & Ausblick

1. Motivation
2. Zugangsschutz
3. Nutzerverhalten

Motivation

- Drahtlose Funknetzwerke heute wohl verbreitetste Art der Netzwerkanbindung
- Übertragung sensibler Informationen
- Physikalische sehr exponiert, (passiver) Angreifer muss lediglich in Reichweite sein
- Entsprechender Schutz der Verbindung auf OSI-Schicht 2 obligatorisch
  - Vertraulichkeit
  - Integrität
  - (Authentizität)

Motivation

- Sicherheit (gemäß genannter Ziele) abhängig von:
  - Aufbau des kryptografischen Verfahrens
  - Komplexität und Länge des geheimen Schlüssels
  - Fehlerfreiheit der technischen Umsetzung/Implementierung (Heartbleed etc.)

Verfahren für den Zugangsschutz

- Ziel: Absichern der drahtlosen Kommunikation auf Schicht 2
- Historie:
  - WEP (Wired Equivalent Privacy) Veraltet und sehr unsicher
  - WPA (Wi-fi Protected Access)
    - “Übergangslösung” bis zur vollständigen Standardisierung von IEEE 802.11i (a.k.a. WPA2)
    - Einsatz schwacher, heute als gebrochen eingestufter Kryptoverfahren wie TKIP auf Basis von RC4
  - WPA2 (Wi-fi Protected Access 2)
    - Verdrängt seit 2004 seine unsicheren Vorgänger

Zugangsschutz: WPA2

- Vollständige Umsetzung von IEEE 802.11i (RSN)
- Unterstützt zwei Authentifizierungsmodi
  - WPA2-PSK: „ Personal“ mit Pre-Shared-Key
  - WPA2-EAP: „Enterprise “ mit 802.1X (EAP over IEEE 802)
- Aushandlung individueller Sitzungsschlüssel (PTK) in 4-Wege-Handshake

- Fokus heute: WPA2-PSK
  - De facto Standard für Heimnetzwerke & kleine Firmennetze

WPA2-PSK: Angriffe möglich?

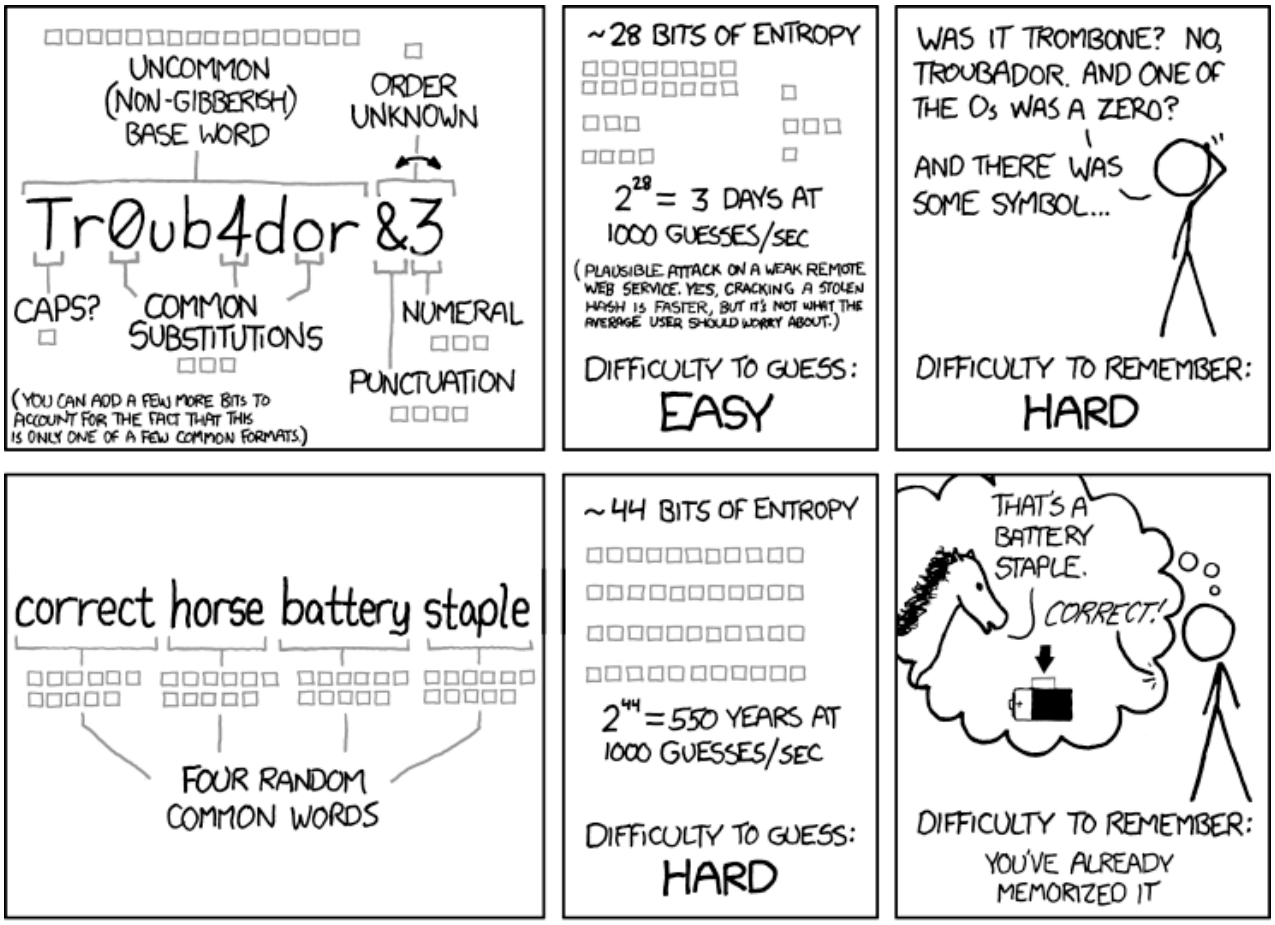
- Angriffe auf Protokollebene (theoretisch) existent, jedoch ...
  - ... starke Vorannahmen
  - ... oftmals nicht praktikabel
- Rückgriff auf Bruteforce-Angriff, um PSK zu ermitteln

- Interessante Alternative: Social-Engineering-Attacken
  - Mensch oft das schwächste Glied bei Sicherheitsmechanismen
  - Nutzer durch geschickte Manipulation zur Preisgabe des Schlüssels bringen
  - Heute nicht Thema, für Interessierte: WifiPhisher, Fluxion

WPA2-PSK: Bruteforce in Praxis realistisch?

- Beständigkeit von WPA2 gegenüber Bruteforce-Angriffen hängt maßgeblich vom verwendeten PSK ab
  - *“Most users will take the road of least resistance.”* [4]
  - Länge Ø 6 Zeichen (min. 8 für PSK bei WPA2)
  - 80% alphabetisch, nur 13.7% alphanumerisch
  - Menschen tendieren dazu, kurze, in Wörterbüchern gelistete Begriffe (oder Kombination daraus) zu verwenden
  - Standardpasswörter sterben nicht aus
    - ISPs und Hardware-Hersteller setzen teils erschreckend schlechte Standardpasswörter

WPA2-PSK: Angriffe in Praxis realistisch?



Quelle: https://xkcd.com/936/

1. Einleitung

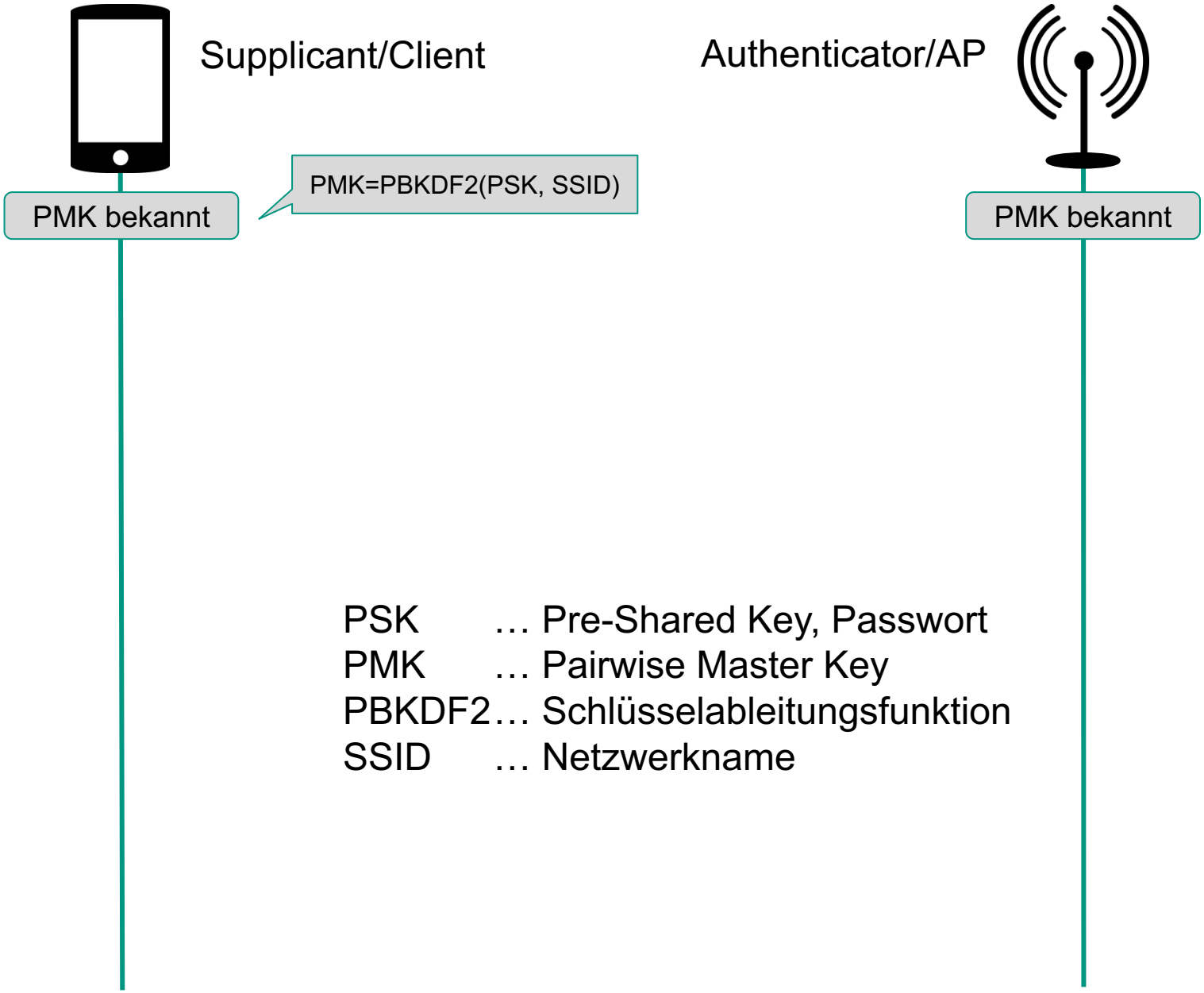
2. WPA2-Cracking

3. Handshakes mitschneiden
4. Fazit & Ausblick

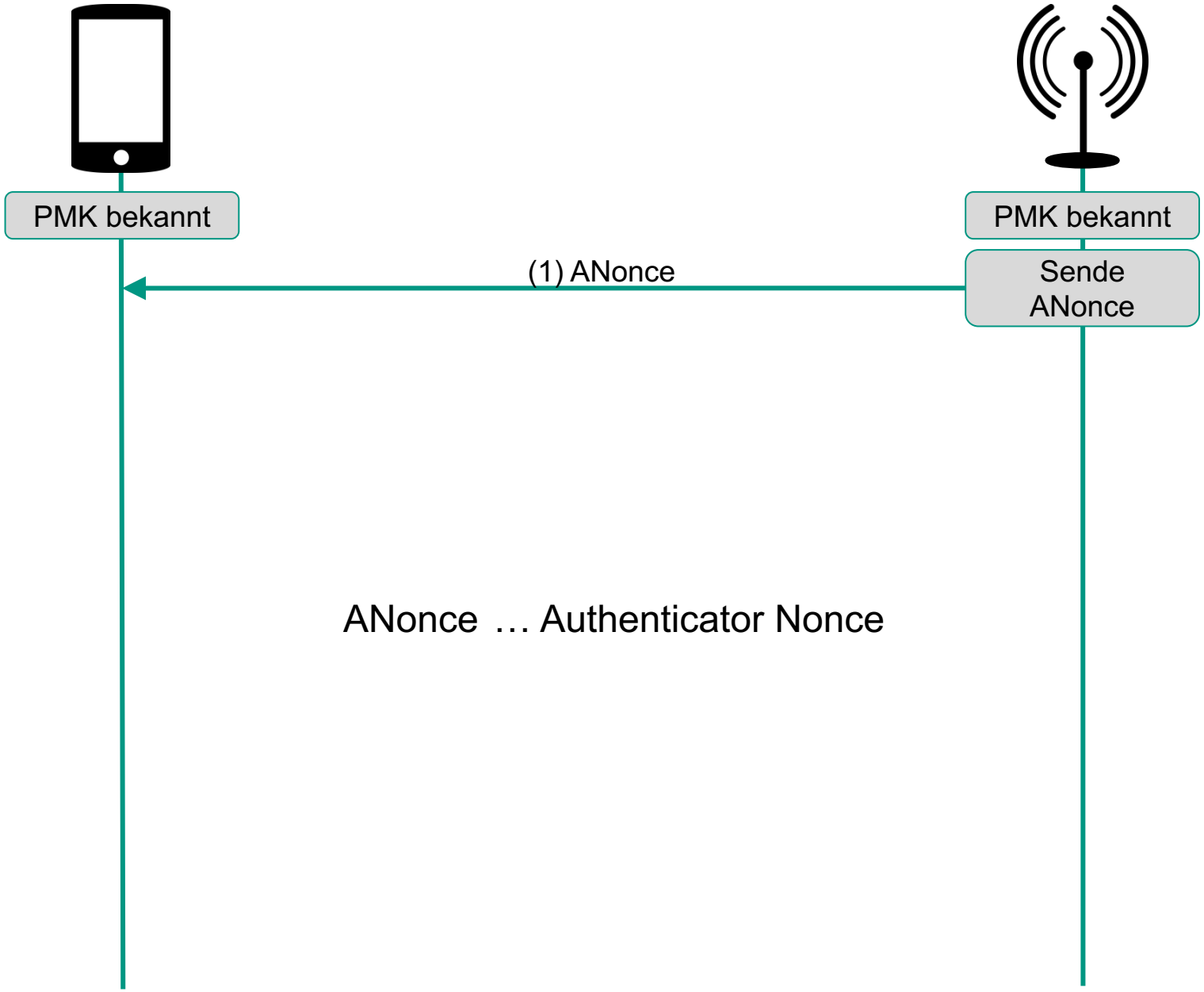
- Der WPA2-Handshake
- Online-Cracking
- Offline-Cracking

Cracking: Viele Wege führen nach Rom...

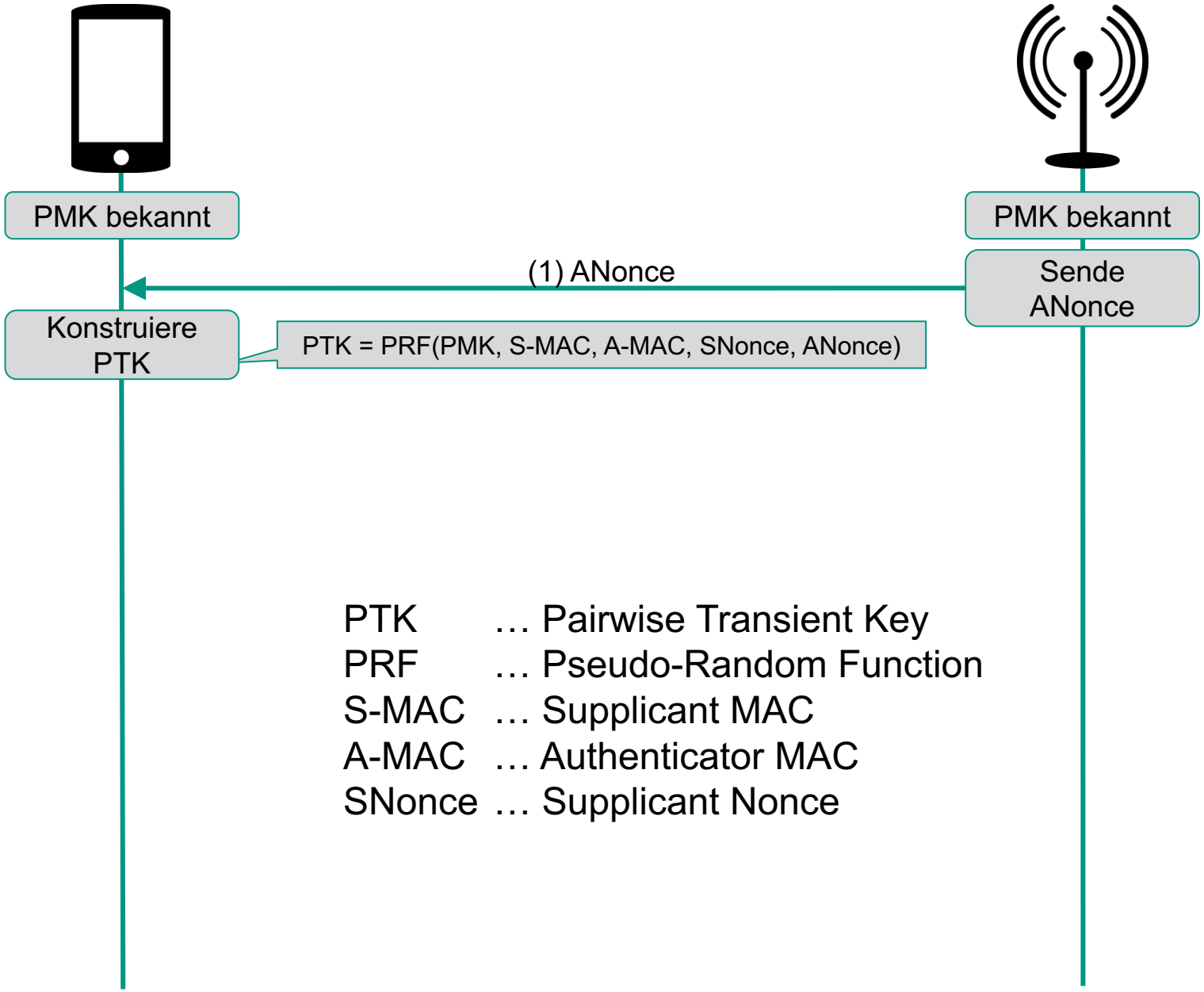
- Zwei generelle Ansätze für Bruteforce-Angriffe:
  - Online
  - Offline
- Für Bruteforce-Angriffe auf WPA2-PSK gilt immer:
  - Passwortkandidaten finden, mit dem Authentifikation beim Access Point möglich ist (Kollision)



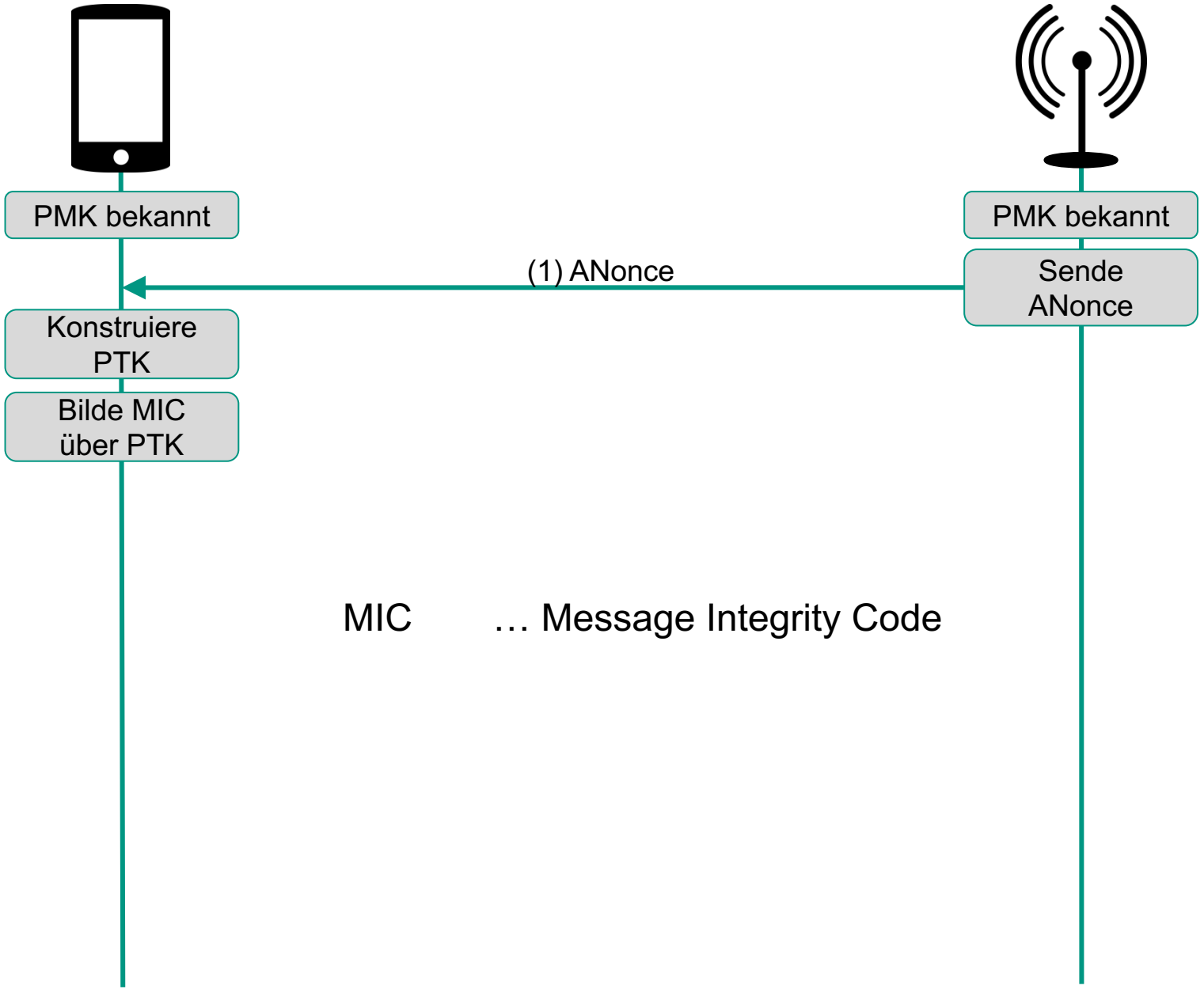
PSK … Pre-Shared Key, Passwort
PMK … Pairwise Master Key
PBKDF2 … Schlüsselableitungsfunktion
SSID … Netzwerkname



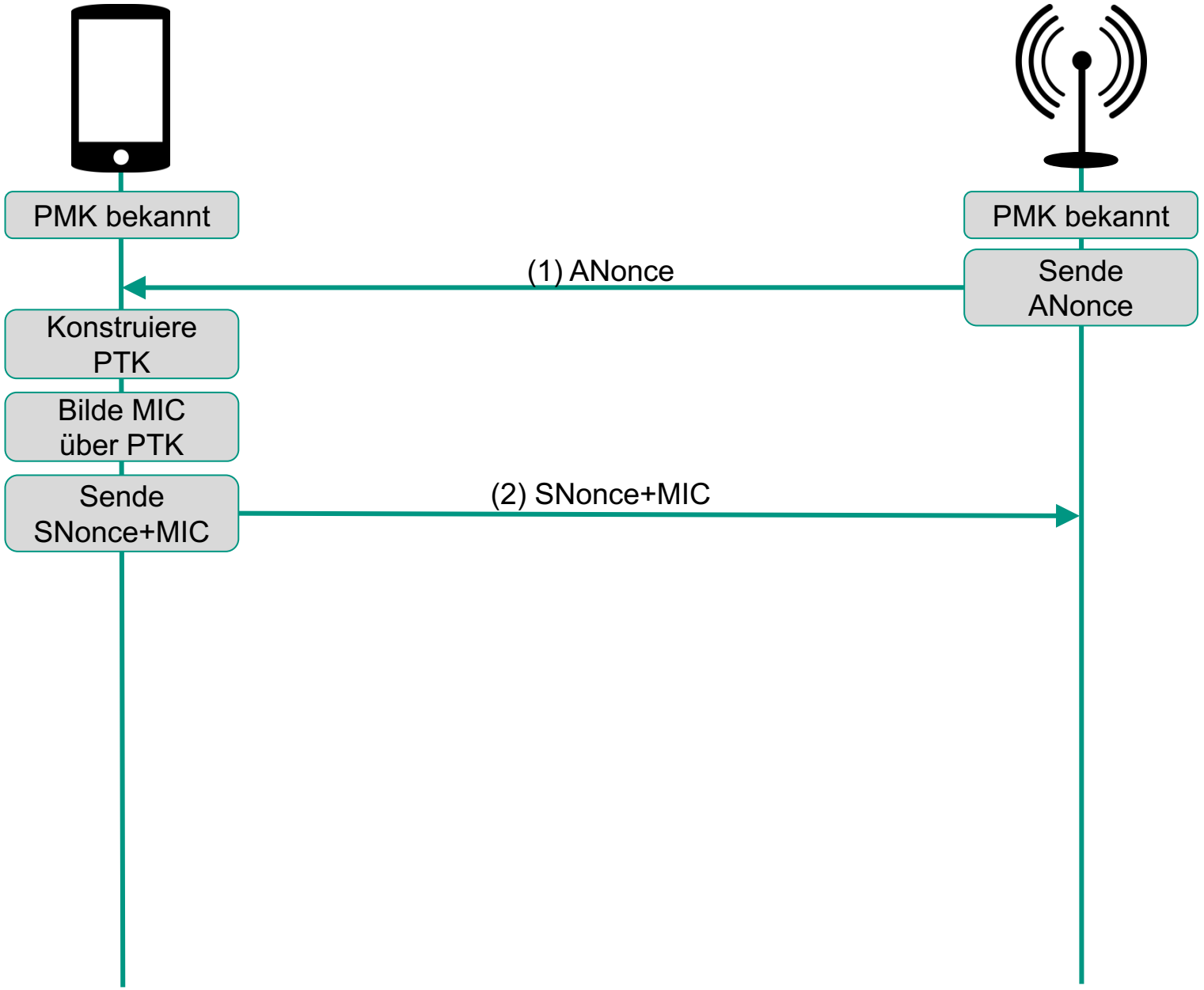
ANonce … Authenticator Nonce

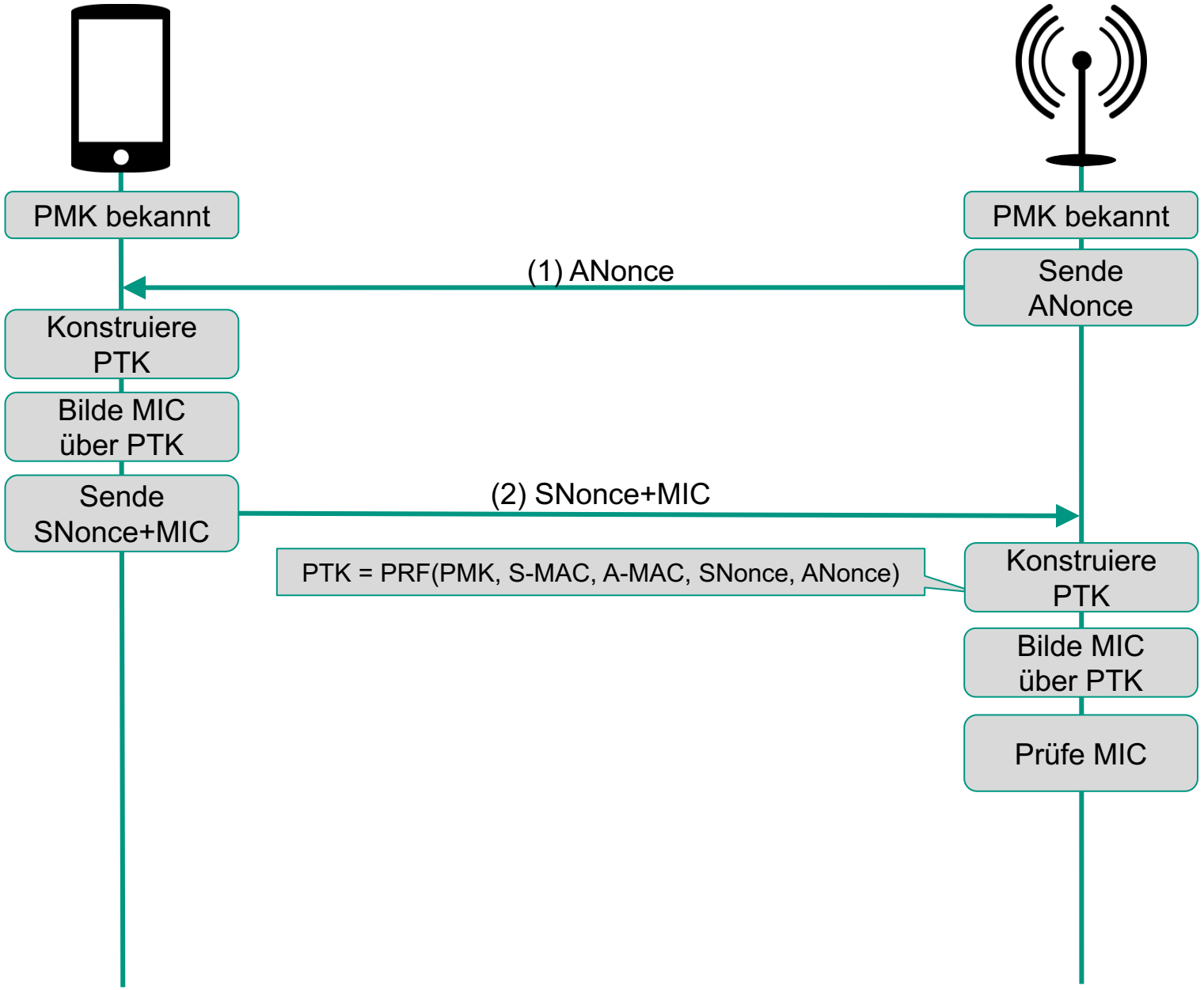


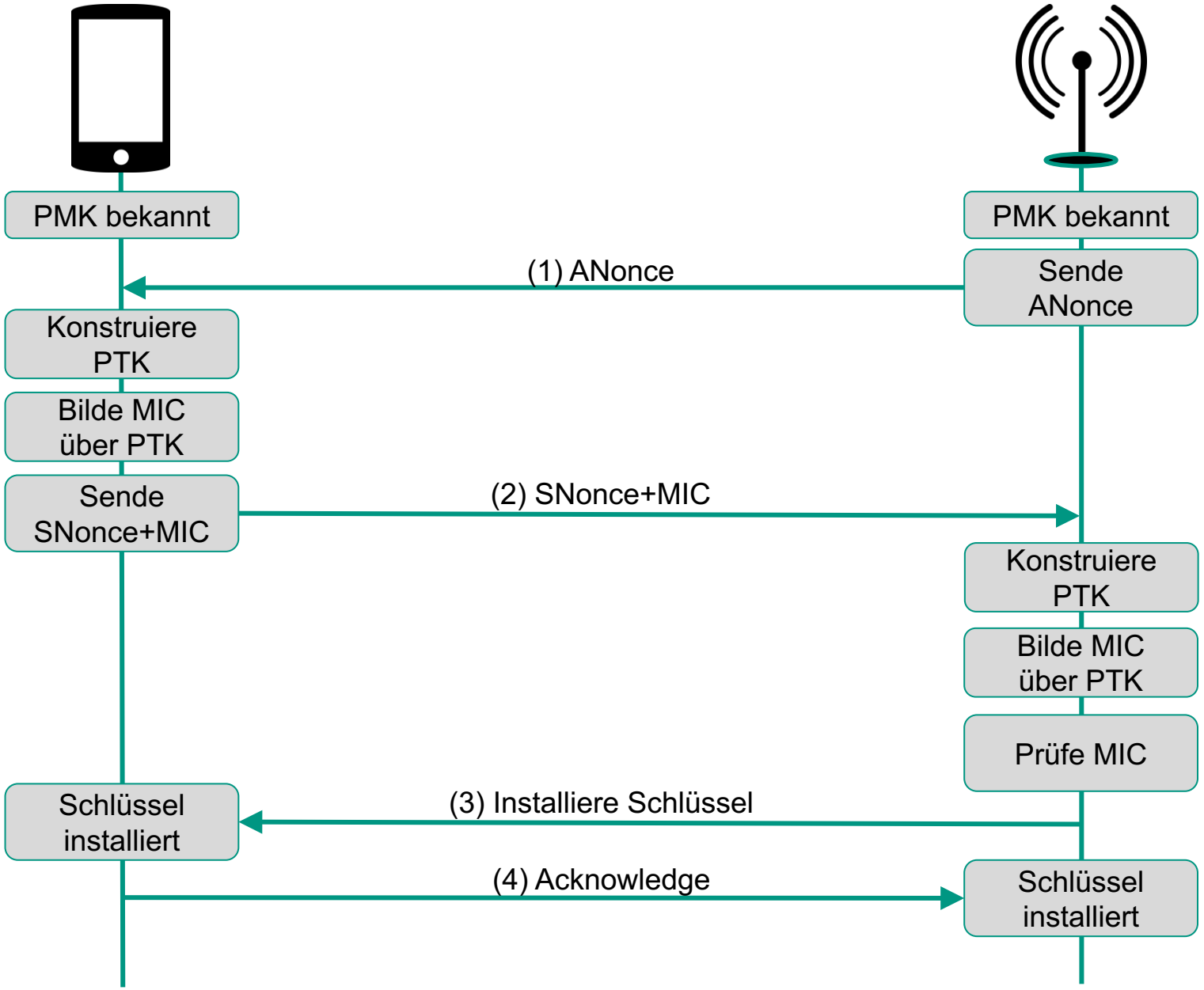
PTK … Pairwise Transient Key
PRF … Pseudo-Random Function
S-MAC … Supplicant MAC
A-MAC … Authenticator MAC
SNonce … Supplicant Nonce



MIC … Message Integrity Code


PMK bekannt
PMK bekannt
(1) ANonce
Sende ANonce
Konstruiere PTK
Bilde MIC über PTK
Sende SNonce+MIC
(2) SNonce+MIC


PMK bekannt
PMK bekannt
(1) ANonce
Sende ANonce
Konstruiere PTK
Bilde MIC über PTK
Sende SNonce+MIC
(2) SNonce+MIC
PTK = PRF(PMK, S-MAC, A-MAC, SNonce, ANonce)
Konstruiere PTK
Bilde MIC über PTK
Prüfe MIC


PMK bekannt
PMK bekannt
(1) ANonce
Sende ANonce
Konstruiere PTK
Bilde MIC über PTK
Sende SNonce+MIC
(2) SNonce+MIC
Konstruiere PTK
Bilde MIC über PTK
Prüfe MIC
Schlüssel installiert
(3) Installiere Schlüssel
(4) Acknowledge
Schlüssel installiert

Cracking: Online

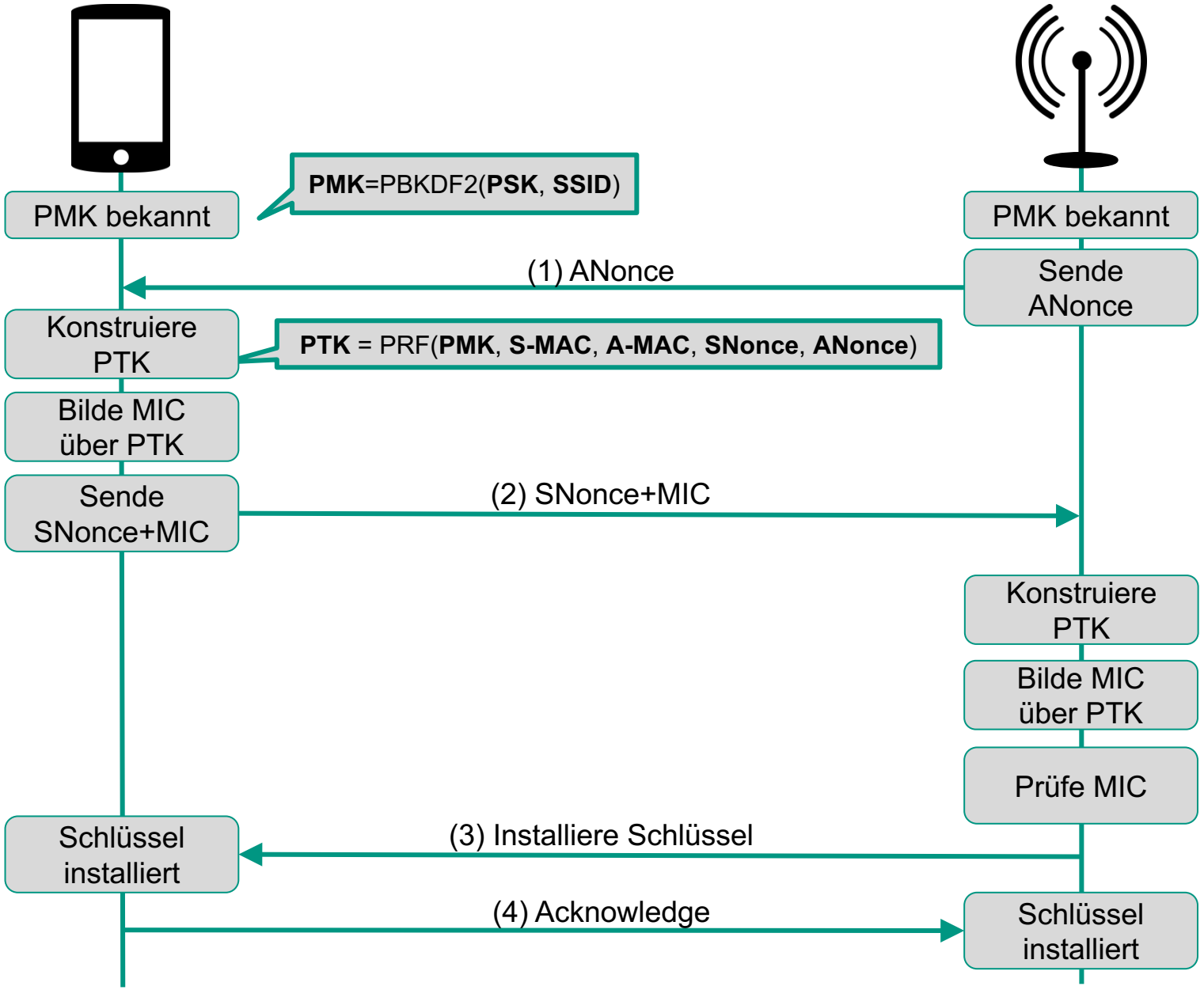
- Annahme: Angreifer in Reichweite von AP
- Angreifer „rät“ das Passwort, gibt sich als Client aus
- Durchführen des Handshakes...
  - ... bis inklusive Schritt 3
  - Wenn MIC von AP akzeptiert und bestätigt → Passwort korrekt
- Nachteile:
  - Mehrere Bottlenecks (AP & Medium) auf die Angreifer keinen Einfluss hat
  - Angriff leicht zu detektieren und bspw. durch Wartezeiten einfach zu verhindern

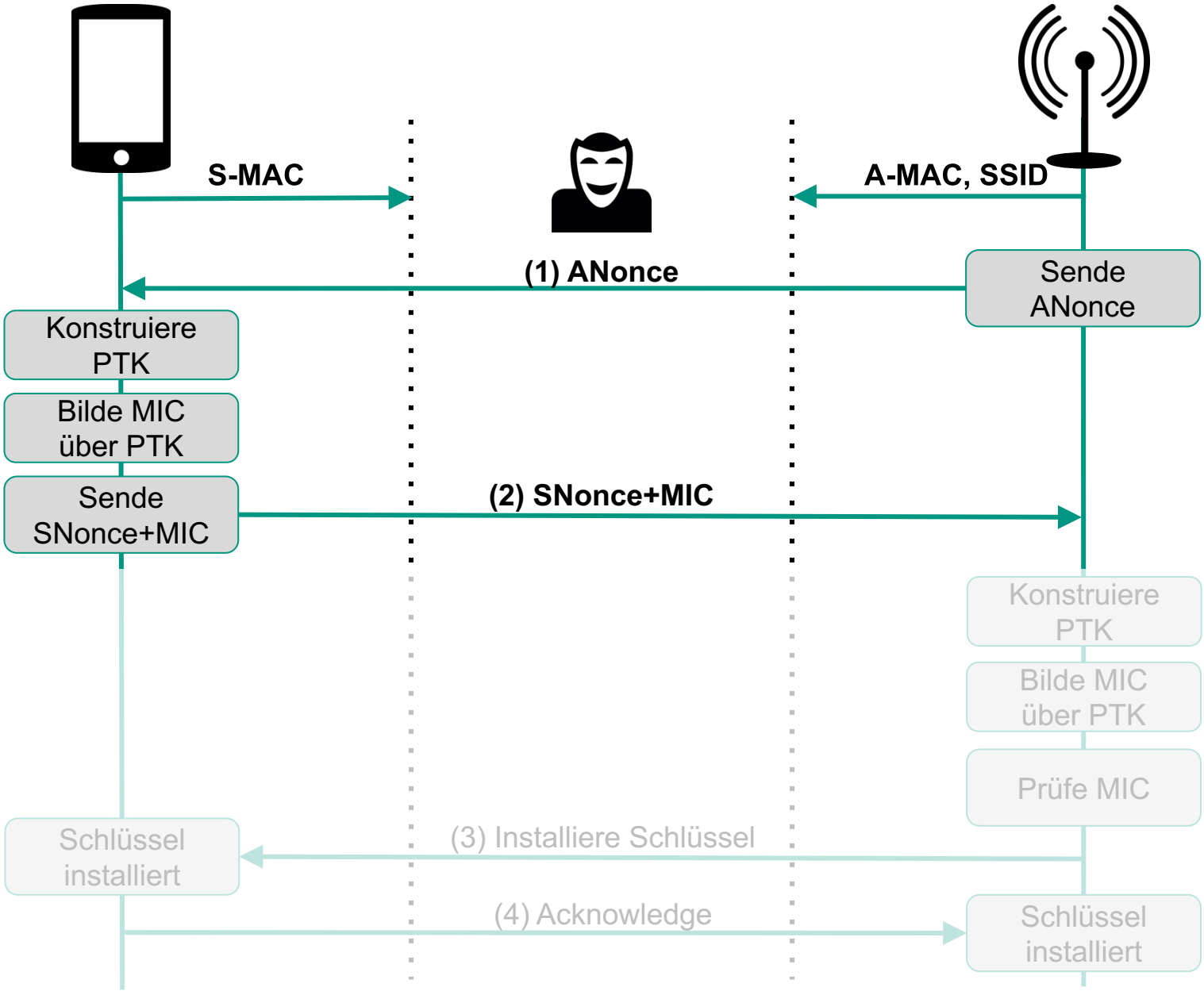
Online-Cracking: Realitätscheck

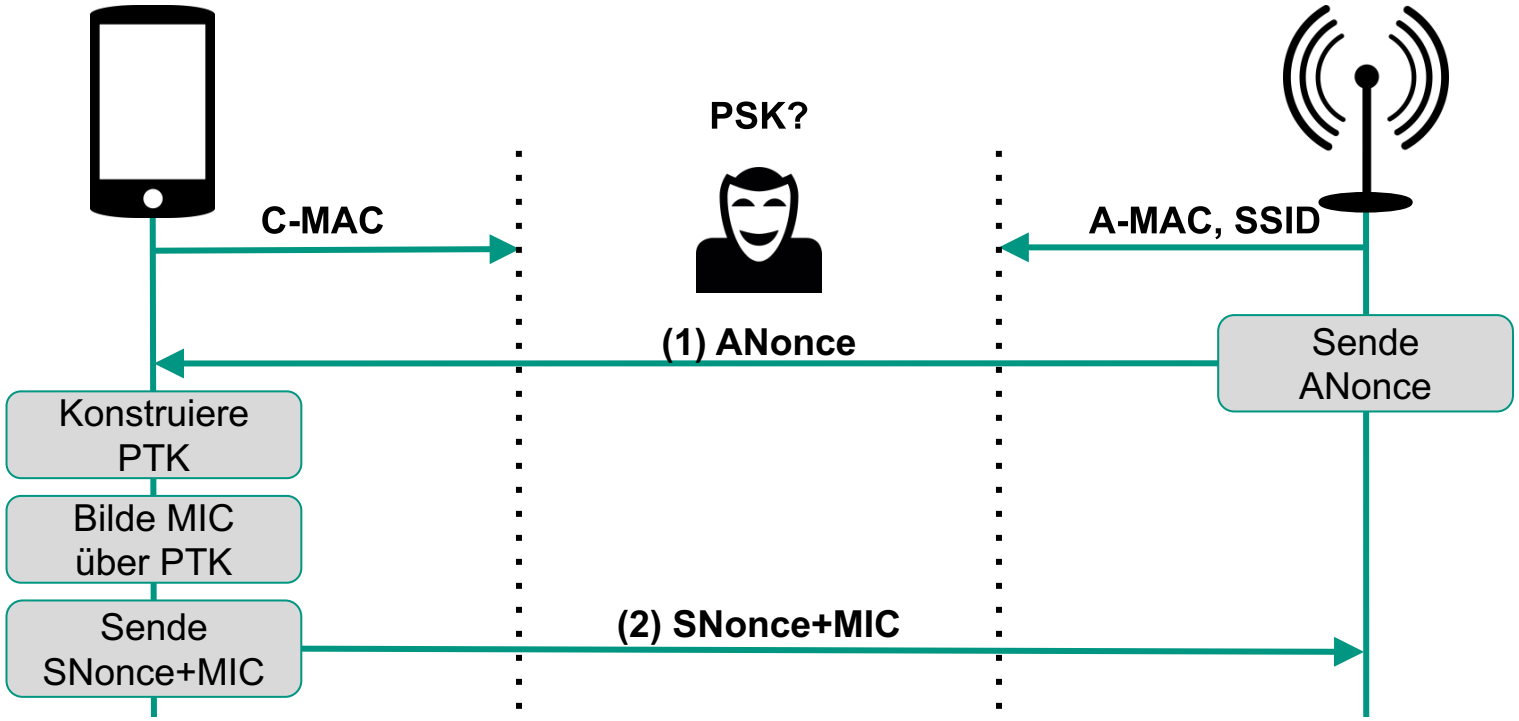
- Annahme:
  - Kompletter Handshake und Verifikation durch Router benötigt lediglich 50 ms, dann 20 Hashes/Sekunde
  - Es sei bekannt, dass das Passwort aus genau 8 Kleinbuchstaben besteht(= 26^8 = 208.827.064.576)
- Hash-Kollision hier nach circa 165 Jahren zu erwarten
- Bedrohung also äußerst gering, Hashes/Sekunde-Rate viel zu limitiert
  - Erfolgschancen verschwindend gering
- Geht das nicht auch effizienter? $\Rightarrow$ Offline-Cracking

Cracking: Offline

- Idee: Angreifer spielt Rolle des AP beim Handshake (später) lokal nach
  - Es werden allerdings einige Informationen benötigt: SSID, SNonce, ANonce usw.
- Woher diese nehmen? Siehe Handshake!


PMK=PBKDF2(PSK, SSID)
PMK bekannt
PMK bekannt
(1) ANonce
Sende ANonce
Konstruiere PTK
PTK = PRF(PMK, S-MAC, A-MAC, SNonce, ANonce)
Bilde MIC über PTK
Sende SNonce+MIC
(2) SNonce+MIC
Konstruiere PTK
Bilde MIC über PTK
Prüfe MIC
Schlüssel installiert
(3) Installiere Schlüssel
(4) Acknowledge
Schlüssel installiert


S-MAC
A-MAC, SSID
(1) ANonce
Sende ANonce
Konstruiere PTK
Bilde MIC über PTK
Sende SNonce+MIC
(2) SNonce+MIC
Konstruiere PTK
Bilde MIC über PTK
Prüfe MIC
Schlüssel installiert
(3) Installiere Schlüssel
(4) Acknowledge
Schlüssel installiert



- Angreifer spielt Handshake jetzt lokal nach
  - Möglichen Wert für PSK bestimmen
  - PMK für diesen berechnen (PBKDF2 → 4096 Runden HMAC-SHA1)
  - Berechneter MIC == aufgezeichneter MIC? → mögliches Passwort gefunden

Cracking: Offline (Tonspur)

- Benötigt werden:
  - Schicht-2-Adressen von Client und Access Point
  - S(upplicant)Nonce und A(uthenticator)Nonce, SSID, MIC
- ⇒ Alle nötigen Informationen nach Schritt 2 übertragen, Schritt 3+4 entbehrlich ⇒ sprich „halber“ Handshake benötigt
- Angreifer spielt Handshake lokal nach
  - Möglichen Wert für PSK bestimmen
  - PMK für diesen berechnen (4096 Runden PBKDF2-Hashfunktion)
  - Berechneter MIC == aufgezeichneter MIC? → mögliches Passwort gefunden

Offline-Cracking: Realitätscheck

- Hardware: Verbund aus 8 x Nvidia GTX 1080 + 2 x Intel Xeon E5 2620V3 leistet 3177,6 kHashes/Sekunde (Sagitta Brutalis, ca. 18500 $) [7]

- Passwort aus vorherigem Beispiel voraussichtlich in 9h gebrochen (statt in 165 Jahren!)
- 8 alphanumerische Zeichen benötigen aber immer noch im Mittel 400 Tage
- Problem: Suchraum gigantisch, Laufzeit der erschöpfenden Suche explodiert

- Geht das nicht auch effizienter? $\Rightarrow$ Wörterbuchangriff

Cracking: Offline mit Wörterbuch

- Motivation: Nutzer favorisieren leicht zu merkende Passwörter
  - *“Users may fulfill policy requirements in predictable ways, such as basing their passwords on names, or words”* [5]
  - Häufig: Simple Kombinationen (bspw. <Wort> + <Zahl>) und Substitutionen (O -> 0, A -> 4) um Richtlinien zu erfüllen
- Statt raten: Passwörter mit Wörterbuch „zusammenbauen“
- Probleme:
  - Güte und Verfügbarkeit von Wörterbuch entscheidend
  - 4096 Runden HMAC-SHA1 pro PMK sind teuer ; kann nur bedingt vorgeneriert werden da von SSID abhängig

Wörterbuch-Cracking: Realitätscheck

- Hardware: Verbund aus 8 x Nvidia GTX 1080 + 2 x Intel Xeon E5 2620V3 leistet 3177,6 kHashes/Sekunde (Sagitta Brutalis, ca. 18500 $) [7]
- Annahmen:
  - Passwort aus max. 2 oft benutzen Wörtern (Top 10.000)
  - enthält bis zu eine Zahl aus maximal 2 Ziffern
  - Ordnung der Einzelteile beliebig
  - = 10.001 * 10.001 * 111 * 3 = 33.306.660.333
- Circa 3 Stunden für Prüfen des vollständigen Suchraums nötig
- Falls man keine entsprechende Hardware besitzt...
  - ... Kapazitäten bei AWS mieten
  - ... Online Services für WPA2-Cracking (Seriosität fraglich)

Offline-Cracking in der Cloud

- Falls man keine entsprechende Hardware besitzt...
  - ... Kapazitäten bei AWS mieten
  - ... Online Services für WPA2-Cracking
    - Bspw.: GPUHash.me
      - GPU-Cluster mit 1,6 MHashes/Sekunde
        4$/erfolgreichem Handshake, 25% Erfolgsrate
    - Viele weitere Services, teilweise gratis
    - Seriosität der Anbieter fraglich

1. Einleitung
2. WPA2-Cracking

3. Handshakes mitschneiden

4. Fazit & Ausblick

- On-Site
- Off-Site

Handshakes (effizient) mitschneiden

- Es existieren zwei Familien von Angriffen
  - On-Site
    - Passives Lauschen, „Abwarten-und-Tee-Trinken“
      - Authentifizierungen eher selten, ineffizienter Angriff
    - Deauth-Angriff
  - Off-Site
    - Evil-Twin-Angriff

- Abstrakt: Schritt 1+2 des Handshakes zwischen Client und AP mitschneiden für späteres Offline-Cracking

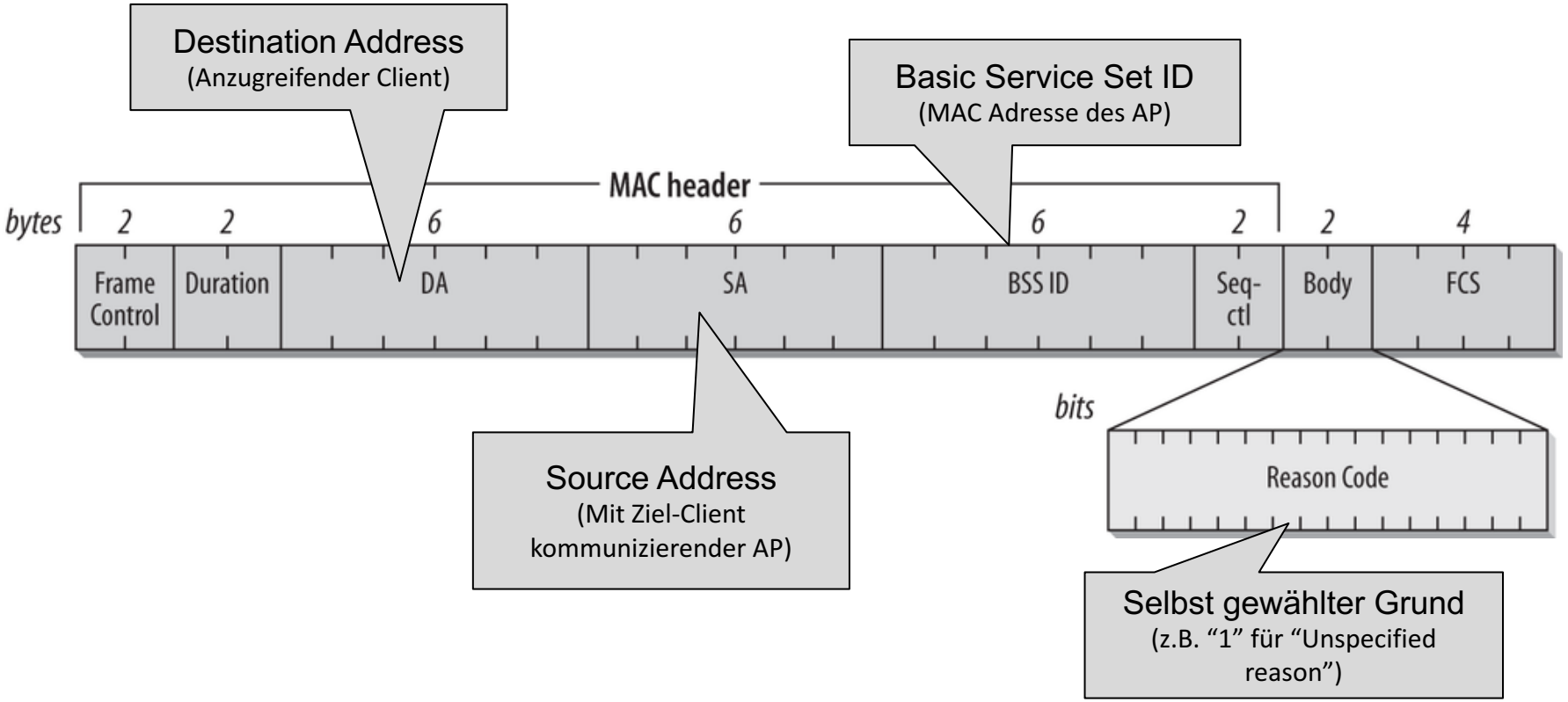
Sniffing: On-Site (passiv)

- Voraussetzung: AP und Client in Reichweite des Angreifers
- Angreifer lauscht auf Medium bis Handshake vorkommt
- Aufzeichnen des Nachrichtenverkehrs
  - Nachträglich auf Handshakes analysieren (.pcap Captures)
  - Live auf dem Interface (BPF einsetzen)
- Nachteile:
  - (Neu-) Authentifizierungen eher selten
  - AP muss in Reichweite sein, nur umgebende Netzwerke und deren Nutzer angreifbar

Sniffing: On-Site (aktiv), aka „Deauth-Attack“

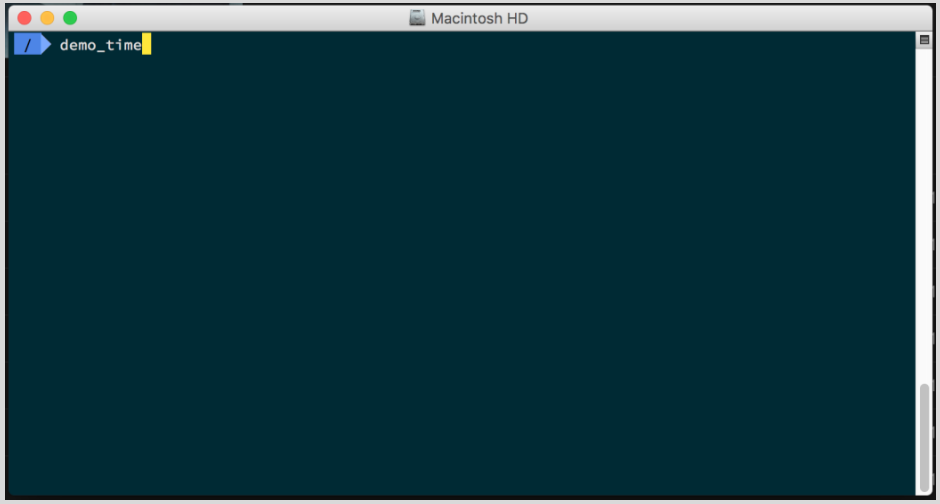
- Idee: Forciere Neu-Authentifizierung des Clients beim AP
- Ausnutzen von Deauthentication-Frames
  - Management-Frames unverschlüsselt; ohne Kenntnis von PSK erzeugbar
  - Angreifer sendet Deauth-Frame an Client (und AP)
  - Gespoofter Absender: MAC-Adresse des AP
  - Experiment zeigt: Funktioniert auch Disassociation-Paketen
- Nachteile:
  - Aktives Eingreifen in den Netzwerkverkehr
  - Je nach Szenario „leicht“ detektierbar, für Nutzer jedoch quasi nicht wahrnehmbar

Sniffing: On-Site (aktiv), aka “Deauth-Attack“



Quelle: https://www.safaribooksonline.com/library/view/80211-wireless-networks/0596100523/httpatomoreillycomsourceoreillyimages1595847.png

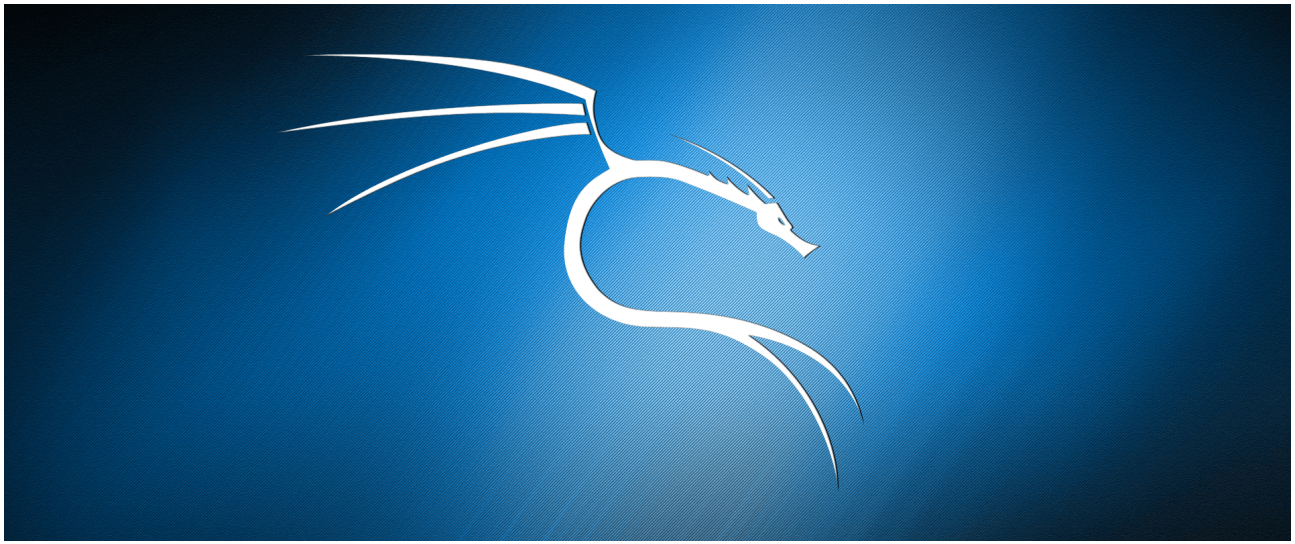
Demo



- Deauth-Angriff:
  - (Ziel auswählen)
  - Deauth-Pakete senden
  - Handshake mitschneiden
  - Handshake mit Wörterbuch brechen

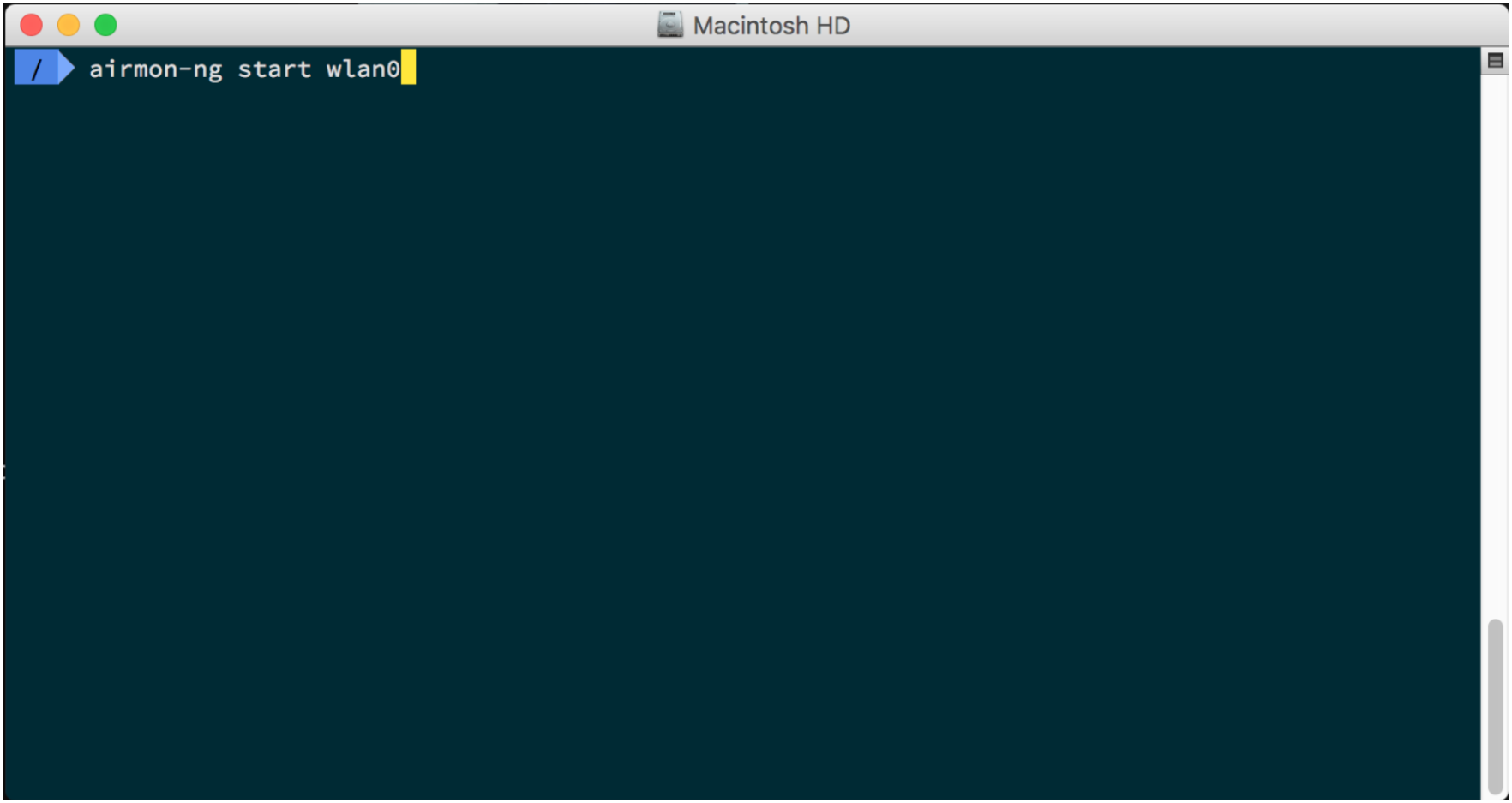
Demo: Eingesetztes System

- Via Parallels virtualisiertes Kali Linux auf MacBook Pro
- Wi-fi-Adapter: TP-Link TL-WN722N mit Realtek rtl8187 Chipsatz



Cracking: Praktische Durchführung

- Monitor-Mode für Netzwerk-Interface aktivieren
  - `airmon-ng start wlan0`
- Aufzeichnen des relevanten Netzwerkverkehrs
  - `airodump-ng --bssid <BSSID> -c <CHANNEL> --write … wlan0mon`
- Handshake provozieren, Client deauthentifizieren
  - `./deauth_jammer.py -a <BSSID> -t <TARGET MAC> -c <CHANNEL> -n 32`
- Verifikation mit `pyrit –r <DUMP> analyze`
- Handshake cracken
  - Hier `Aircrack-ng`
  - *Weitere Tools: `Pyrit`, `Hashcat`, `Reaver`, `John The Ripper`, `CoWPAtty`, `Aircrack-ng`, `oclHashcat/cudaHashcat`*


Macintosh HD
/
airmon-ng start wlan0

„Deauth-Attack“: Realitätscheck

- Wie leicht zu erkennen?

- Feldversuch 1: WPA2-PSK Heimnetzwerk mit bis zu 8 aktiven Clients
  - Über 24 Stunden kein Deauth-Paket
    - Nur ein AP → keine Clientverwaltung notwendig
    - Keine Schlüsselneuaushandlung benötigt
  - Deauth-Attacke äußerst auffällig durch Spikes

„Deauth-Attack“: Realitätscheck

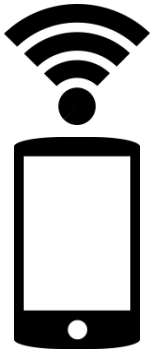
- Feldversuch 2: WPA2-EAP als Firmennetzwerk mit bis zu 30 aktiven Clients
  - Ca. 2 Deauth-Pakete/min, oftmals in Spikes (5-10 Pakete „zeitgleich“)
    - 3 APs in Reichweite → Clientverwaltung notwendig
    - Clients bewegen sich („Reasoncode 8“)
    - Stationäre Systeme benötigen Schlüsselneuaushandlung („Reasoncode 2“)
  - Auch hier: Deauth-Attacke auffällig durch (höhere) Spikes

Sniffing: Off-Site-Angriffe

- Ziel:
  - Angriffe gegen Client über Netzwerke, welche sich nicht in Reichweite befinden
  - Angriffsziele sollen nicht mehr von umgebenden Netzwerken sondern Clients in Reichweite abhängig sein
- Idee: Client zu Authentifizierungsversuch verleiten; Handshake provozieren
- Voraussetzungen:
  - Client muss sich weiterhin in Reichweite befinden
  - Konfiguration (SSID & Sicherheitskonfiguration) des Zielnetzwerkes muss bekannt sein
    - Woher erfährt der Angreifer die SSIDs die dem „Opfer“ bekannt sind ?

Exkurs: Probe-Requests und Probe-Responses

- APs senden in Intervallen sog. „Beacon-Frames“ aus, um Netzwerk und Konfiguration bekannt zu machen
- Client kann Probe-Requests versenden
  - zwingend für Detektion von „versteckten Netzen“
  - enthält SSID
  - AP antwortet mit Probe-Response
- Probe Response enthält u.a.:
  - Unterstützte Datenraten des APs
  - Sicherheitskonfiguration (z.B. OPN, WPA2 PSK)



Exkurs: Probe-Requests und Probe-Responses

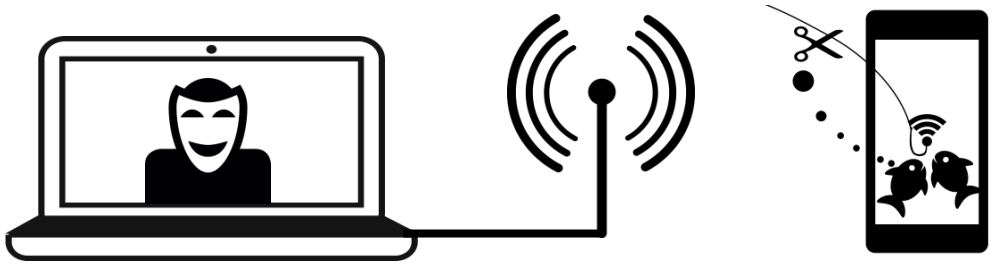
- Viele Geräte senden fortlaufend Probe-Requests aus...
  - Eigene Analysen:
    - Windows Phone, Windows 10, macOS
    - Android 6 sendet auch wenn verbunden!
  - ... auch für nicht versteckte Netze - unnötig und unsinnig!
- Problem dabei:
  - Öffnet Tracking von Endgeräten Tür und Tor
  - Client verrät ihm bekannte Gegenstellen

Sniffing: Off-Site: „Evil-Twin-Attack“

- Ist original AP nötig für validen Handshake? Nein!
  - Liefert lediglich ANonce und SSID
    - ANonce → beliebige Zufallszahl
    - SSID → bekannt aus Probe-Request
  - AP muss sich während WPA2-PSK-Handshake selbst nicht authentifizieren

Sniffing: Off-Site: „Evil-Twin-Attack“

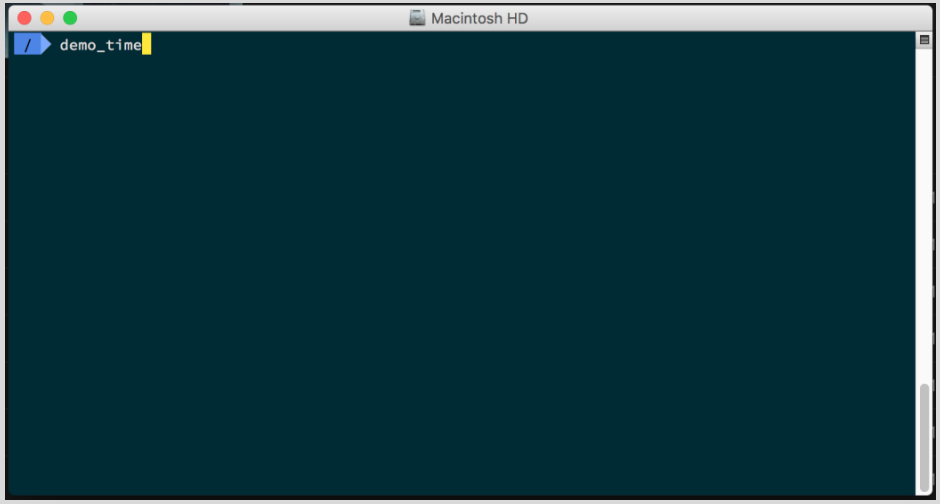
- Erstelle eigenen Mimikry-AP mit SSID aus Probe-Request
  - Sende Beacon-Frames bzw. entsprechende Probe-Response an Client
- Wenn Client „ anbeißt“, dann Handshake bis Schritt 2 durchführen
- Anschließend Verbindungsabbruch
  - Agieren als MITM unmöglich, würde Kenntnis des PSK erfordern



Sniffing: Off-Site: „Evil-Twin-Attack“

- Nachteile:
  - Client muss in Reichweite sein
  - Fake-AP auffällig → aktives Teilnehmen im Netzwerk
- Abhängig vom Endsystem wird Fake-AP “erkannt”
  - Windows-Phone/Windows 10: Erkennt Abweichung des Profils
  - Android 5/6, Windows 8.1: Verbindet automatisch, manche Geräte wechseln sogar bei aktiver Verbindung wenn Signal stärker
- Lösung: Mehrere Konfigurationen auf dem Interface öffnen

Demo



- Evil-Twin-Angriff:
  - (Ziel auswählen)
  - (Probe-Requests auslesen)
  - Evil-Twin aufsetzen
  - Handshake mitschneiden
  - PSK brechen

Evil-Twin-Angriff: Befehle

- `airmon-ng check kill` *// andere, interferiernde Prozesse beenden*
- `airmon-ng start <interface>` *// in der Regel "wlan0"*
- `airodump-ng <interface>` *// Probe Requests mitlesen, Ziel wählen*
- `airbase-ng --essid <ESSID> -Z 4 -F dump <interface>` *// Fake-AP bereitstellen (interface in Monitoring-Mode, siehe Ergebnis von Schritt 2. AP verwendet WPA2-PSK und schreibt Traffic in PCAP-Datei*
- `pyrit -r <dump.pcap> analyze` *// Prüfen, ob gültiger Handshake in dump enthalten*
- `./combinr.py -n 2 wordlist.txt | aircrack-ng -e <ESSID> - w - <dump.cap>`
  // aircrack-ng versucht die generierten, über StdIn hineingereichten, Passwort-Kandidaten durch

1. Einleitung
2. WPA2-Cracking
3. Handshakes mitschneiden

4. Fazit & Ausblick

- Fazit
  - WPA2
  - Vorgestellte Angriffe
- Ausblick

Fazit: WPA2-PSK

- Management-Frames sind nicht (implizit) authentifiziert
  - Erweiterung 802.11w erlaubt Authentifizierung einiger Management Frames
- Keine Perfect-Forward-Secrecy
- AP muss sich gegenüber dem Client bei Handshake nicht authentifizieren
- Probe-Requests kritisch bzgl. Tracking von Endgeräten
  - Erkenntnis: Endgeräte zu gesprächig
  - Ø 50 Probe-Requests/Stunde pro Gerät [9]
    - Unsere Ergebnisse: Spitzen von 40 Probe-Requests/Stunde (Android 6)
  - Android sendet mehr Probe-Requests als iOS, Mac OS, Windows 10

Fazit: Praktikabilität vorgestellter Angriffe

- Komplexität des Schlüssel entscheidet letztlich über Sicherheit von WPA2-Netzen
  - Schwache Schlüssel stellen eine realistische Sicherheitsbedrohung dar
  - Hinreichend komplexe Schlüssel nicht in annehmbarer Zeit zu brechen
- Netze müssen für Angriff auf deren PSK nicht in Reichweite sein
- Vorgestellte On- und Off-Site-Verfahren stellen in Kombination mit Wörterbüchern einen praktikablen Angriff auf den PSK dar

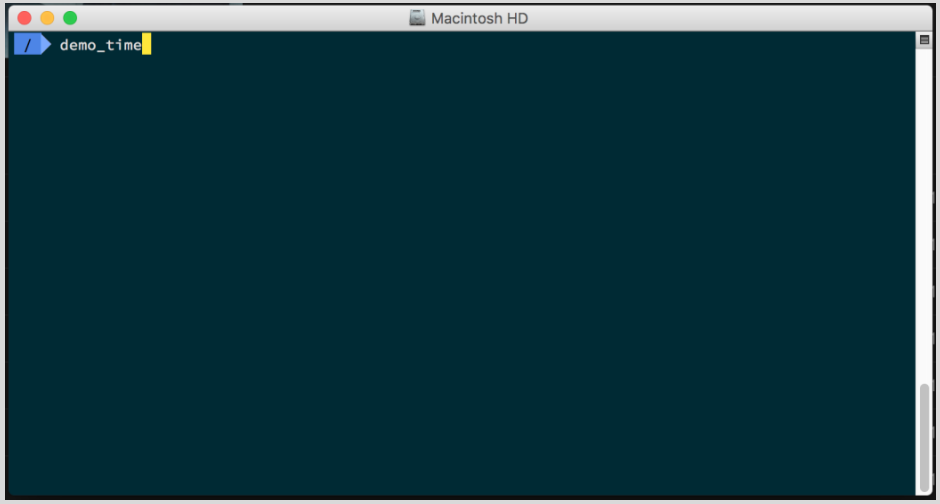
Fazit: Praktikabilität vorgestellter Angriffe

- Detektion von Deauth-Attacken:
  - Normalerweise: AP verwirft Pakete die nicht an ihn gerichtet sind
  - AP erhält Deauth-Paket von „sich selbst“ → möglicher Angreifer im Netz!
  - Nutzen fragwürdig, Clients authentifizieren sich trotzdem
- Prävention von Evil-Twin-Attacken:
  - MAC Randomization erschwert gezielte Angriffe macht sie jedoch nicht unmöglich
  - Speicherung von Netzwerkprofilen zum Erkennen von Evil-Twins bei „falscher“ Konfiguration
  - Speicherung von MAC-Adressen bekannter APs
    - neuer unbekannter AP → Nutzer benachrichtigen
    - Aufwändig für Clients bei vielen APs

Ausblick

- Ein Verbesserungsansatz: Automatisierung des Evil-Twin-Angriffs
  - Horizontale Skalierung:
    - Für alle Probe-Requests eines Clients falsche APs bereitstellen
    - Viele Handshakes mitschneiden
    - Jeden Handshake lediglich gegen eine effiziente Wortliste prüfen
    - “Jedes Smartphone war einmal mit einem schlecht gesicherten Café-WLAN verbunden”
    - $\Rightarrow$ Suche nach der “lowest hanging fruit”

Demo



- Vorstellung von „TINDAHR“

Karlsruher Institut für Technologie

Fragen?

KASTEL

Vielen Dank für Eure Aufmerksamkeit!

Benny Görzig (bgoerzig@gmail.com) & Florian Loch (me@fdlo.ch)

https://github.com/kastel-wpa2

Literatur & Quellen

[1] Jörg Rech. Wireless LANs : 802.11-WLAN-Technologie und praktische Umsetzung im Detail; 802.11a/h, 802.11b, 802.11g, 802.11i, 802.11n, 802.11d, 802.11e, 802.11f, 802.11s, 802.11ac, 802.11ad.

[2] SANS Institute. IEEE 802.11 Pocket Reference Guide.

[3] IEEE Computer Society: IEEE Std 802.11TM-2012 (Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications). 2012.

[4] Moshe Zviran: Password Security: An Empirical Study. 1999. URL: http://calhoun.nps.edu/bitstream/handle/10945/40319/haga_password_security.pdf?sequence=1

[5] Richard Shay. Correct horse battery staple: Exploring the usability of system-assigned passphrases. 2012. URL: https://cups.cs.cmu.edu/soups/2012/proceedings/a7_Shay.pdf

Literatur & Quellen

[6] J. Bonneau, E. Shutova. Linguistic properties of multi-word passphrases. 2012. URL: http://www.jbonneau.com/doc/BS12-USEC-passphrase_linguistics.pdf

[7] Benchmark von Sagitta Brutalis. URL: https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40

[8] Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms. URL: http://papers.mathyvanhoef.com/asiaccs2016.pdf

[9] Short: How Talkative is your Mobile Device? An Experimental Study of Wi-Fi Probe Requests

https://frdgr.ch/wp-content/uploads/2015/06/Freudiger15.pdf

Bildquellen

https://openclipart.org/detail/3619/hammer Icon made by alst

https://openclipart.org/detail/262305/smartphone Icon made by ciubotaru

http://www.flaticon.com/free-icon/anonymous_14446 Icon made by Picol

https://openclipart.org/detail/17423/wirelesswifi-symbol Icon made by ispyisail

https://openclipart.org/detail/208540/laptop Icon made by jmlrtinez

https://de.wikipedia.org/wiki/Aircrack#/media/File:Aircrack-ng-new-logo.jpg

https://www.kali.org/wp-content/uploads/2015/05/kali-dragon-middle.png

https://de.wikipedia.org/wiki/Heartbleed#/media/File:Heartbleed.svg