WiFuzz: Detecting and Exploiting Logical Flaws in the Wi-Fi Cryptographic Handshake

Mathy Vanhoef - @vanhoefm

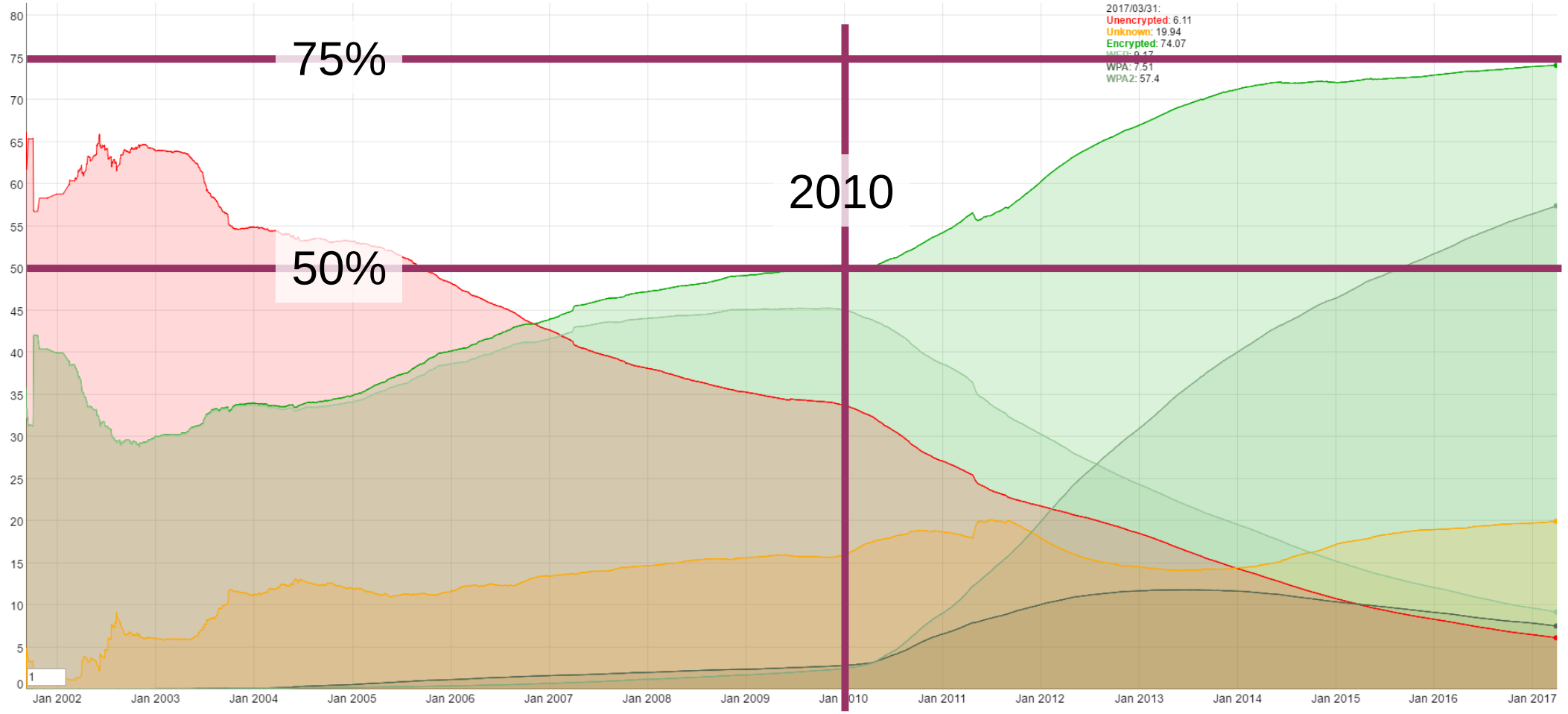
Black Hat, 27 July 2017

In collaboration with Domien Schepers and Frank Piessens

DistriNet

Introduction

More and more Wi-Fi network use encryption:



Most rely on the Wi-Fi handshake to generate session keys

How secure is the Wi-Fi handshake?

Design: formally analyzed and proven secure[1]

Security of implementations?

- Some works fuzz network discovery stage[2]
- Many stages are not tested, e.g. 4-way handshake.
- But do not tests for logical implementation bugs

→ Objective: test implementations of the full Wi-Fi handshake for logical vulnerabilities

[1] C. He, M. Sundararajan, A. Datta, A Derek, and J. Mitchell. A modular correctness proof of IEEE 802.11i and TLS.
[2] L. Butti and J. Tinnes. Discovering and exploiting 802.11 wireless driver vulnerabilities.

Background: the Wi-Fi handshake

Main purposes:

- Network discovery
- Mutual authentication & negotiation of pairwise session keys
- Securely select cipher to encrypt data frames

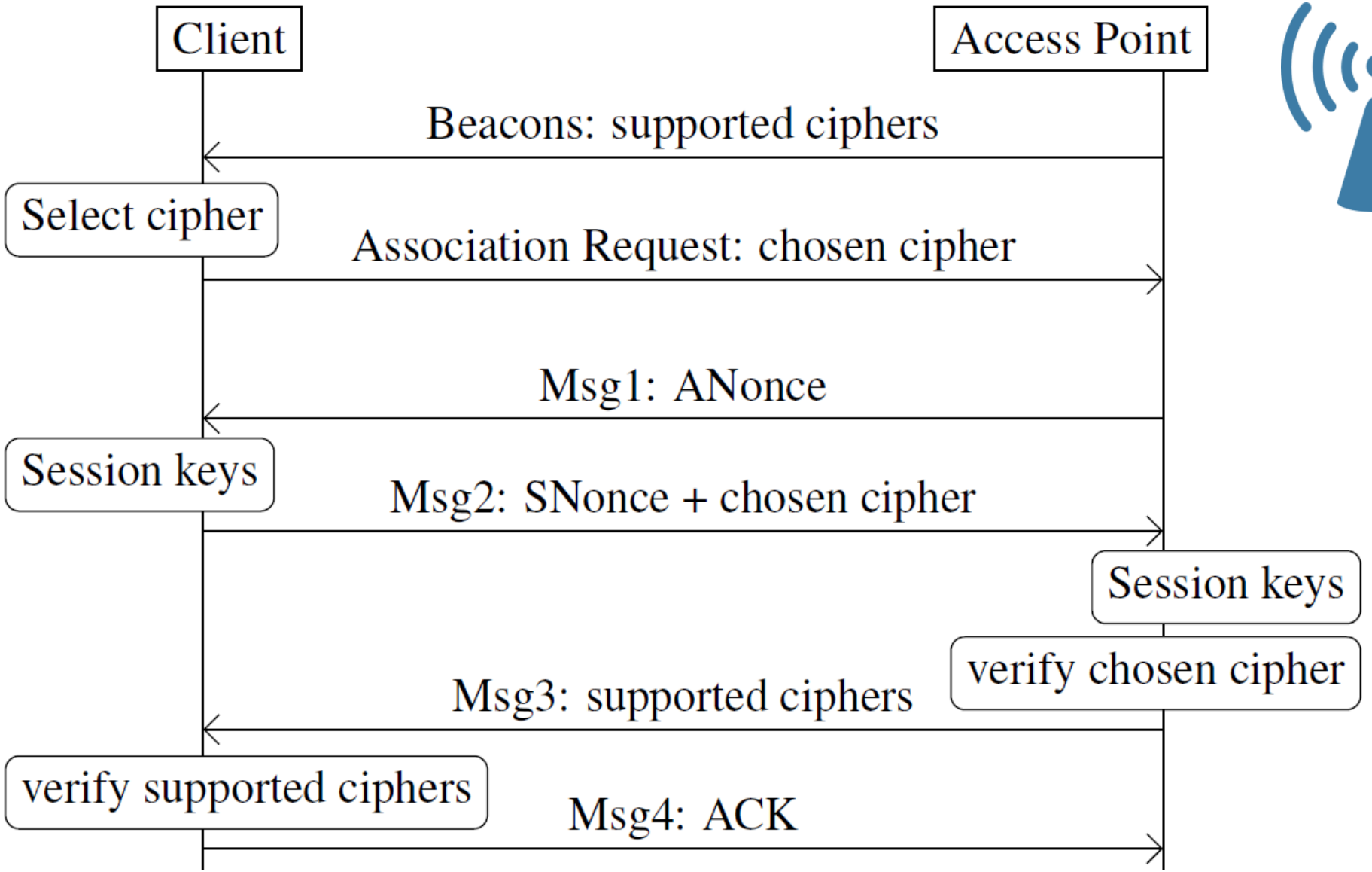
WPA-TKIP

Short-term solution: reduced security so it could run on old hardware

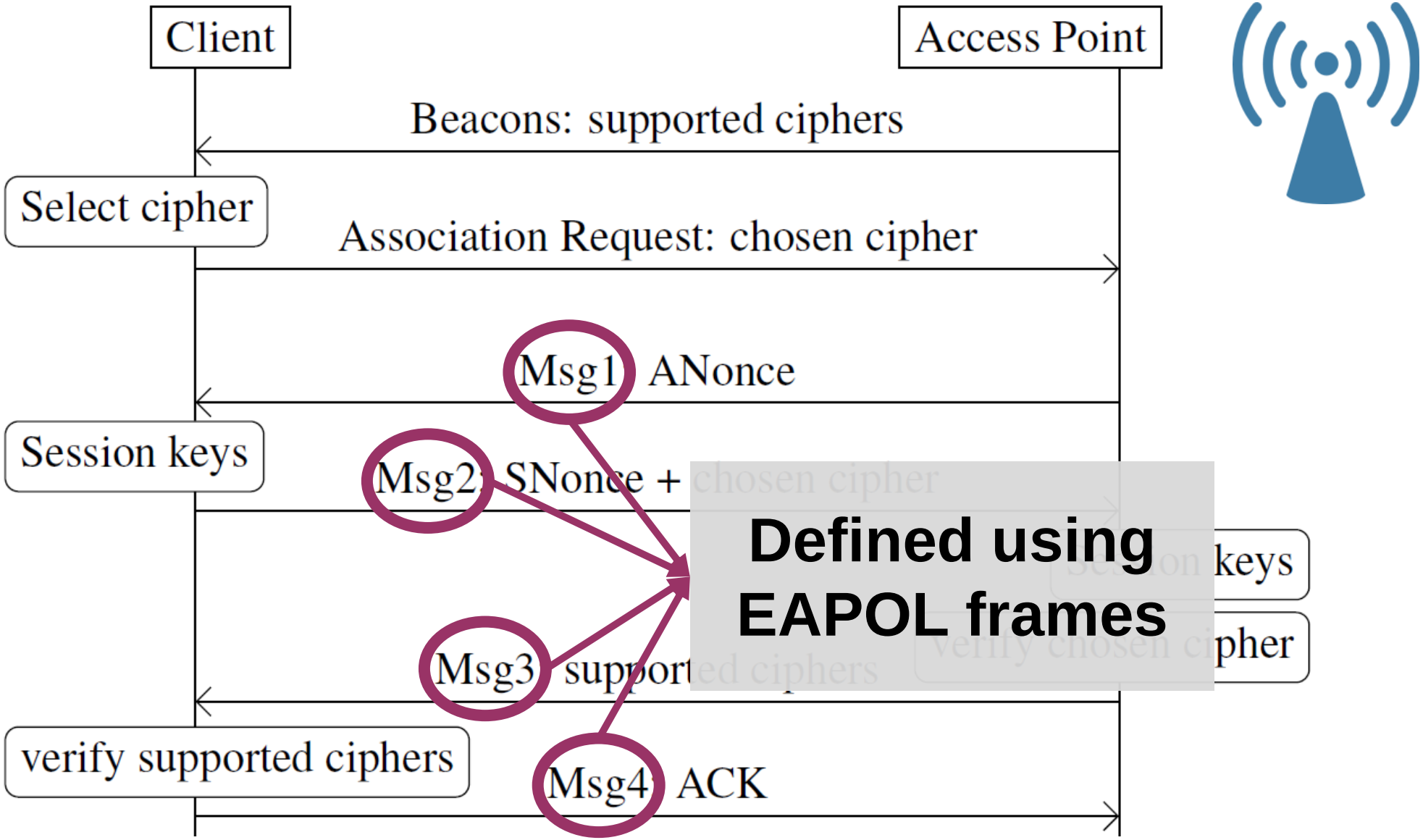
AES-CCMP

Long-term solution based on modern cryptographic primitives

Wi-Fi handshake (simplified)

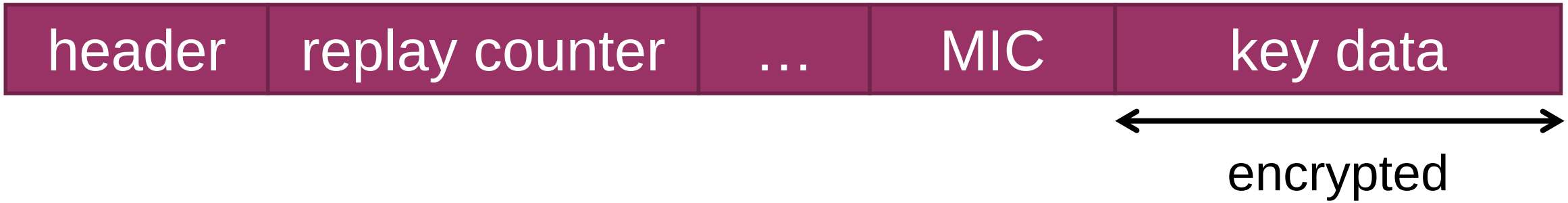


Wi-Fi handshake (simplified)

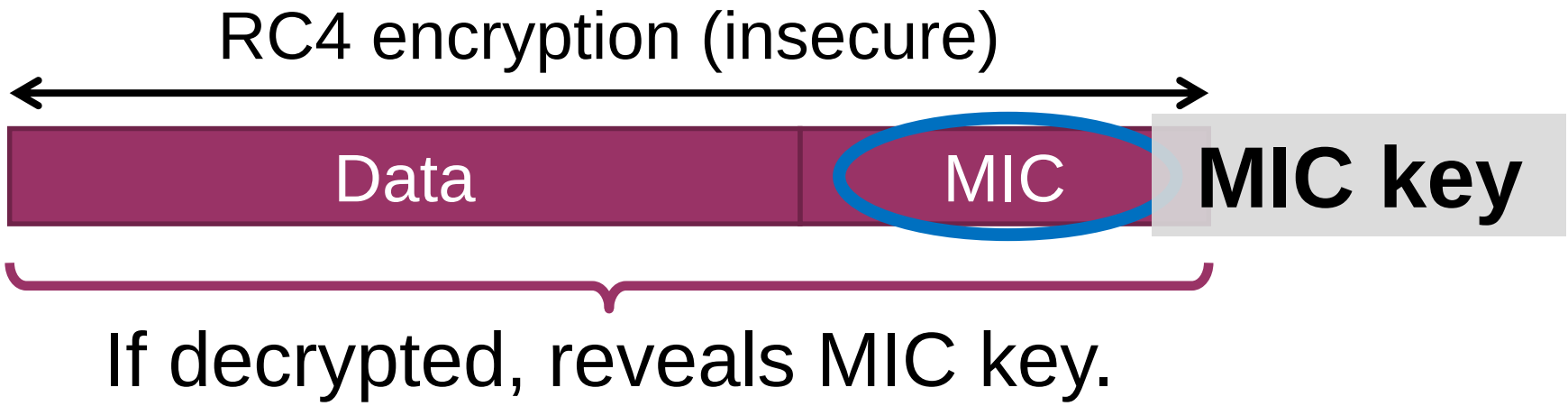


Frame Layouts

- EAPOL frame:



- WPA-TKIP frame:



How to test implementations?

Model-based testing!

- Test if program behaves according to some abstract model
- Proved successful against TLS

  - Apply model-based approach on the Wi-Fi handshake

Model-based testing: our approach

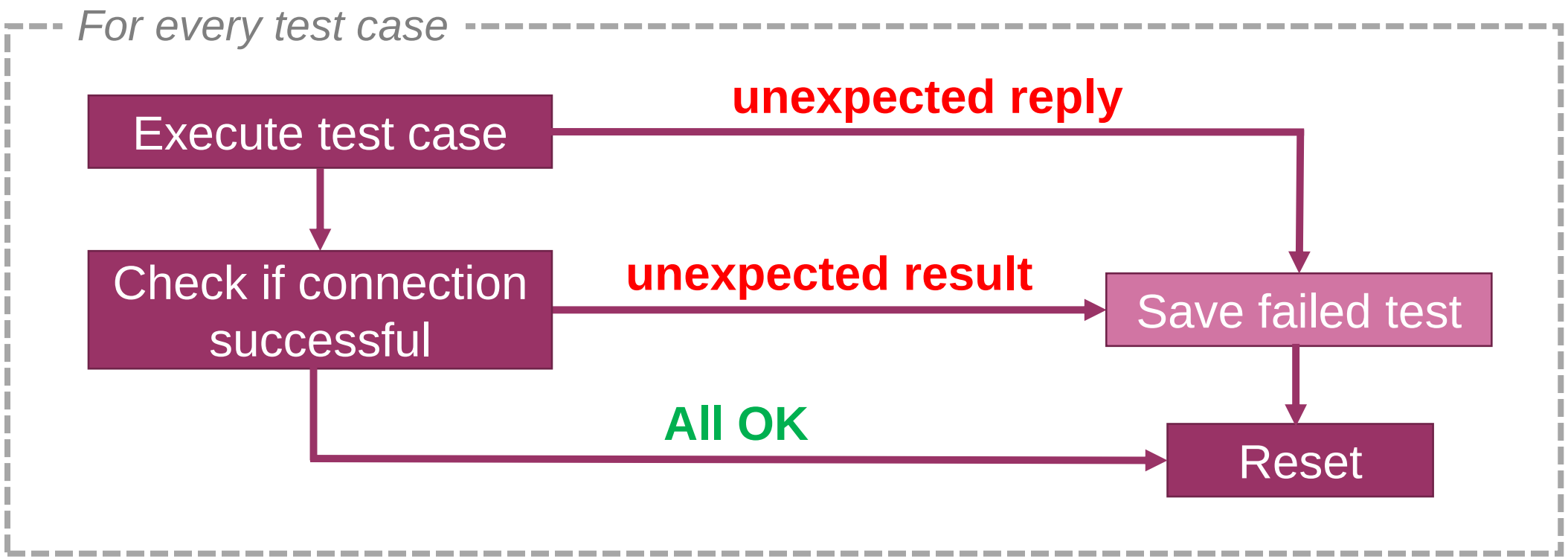
Test generation rules:

- Test various edge cases, allows some creativity
- Are assumed to be independent (avoid state explosion)

A test case defines:

1. Messages to send & expected replies
2. Results in successful connection?

Executing test cases



Afterwards Inspect failed test cases

- Experts determines impact and exploitability

Test generation rules

Test generation rules manipulating messages as a whole:

1. Drop a message
2. Inject/repeat a message

Test generation rules that modify fields in messages:

1. Bad EAPOL replay counter
2. Bad EAPOL header (e.g. message ID)
3. Bad EAPOL Message Integrity Check (MIC)
4. Mismatch in selected cipher suite
5. …

Evaluation

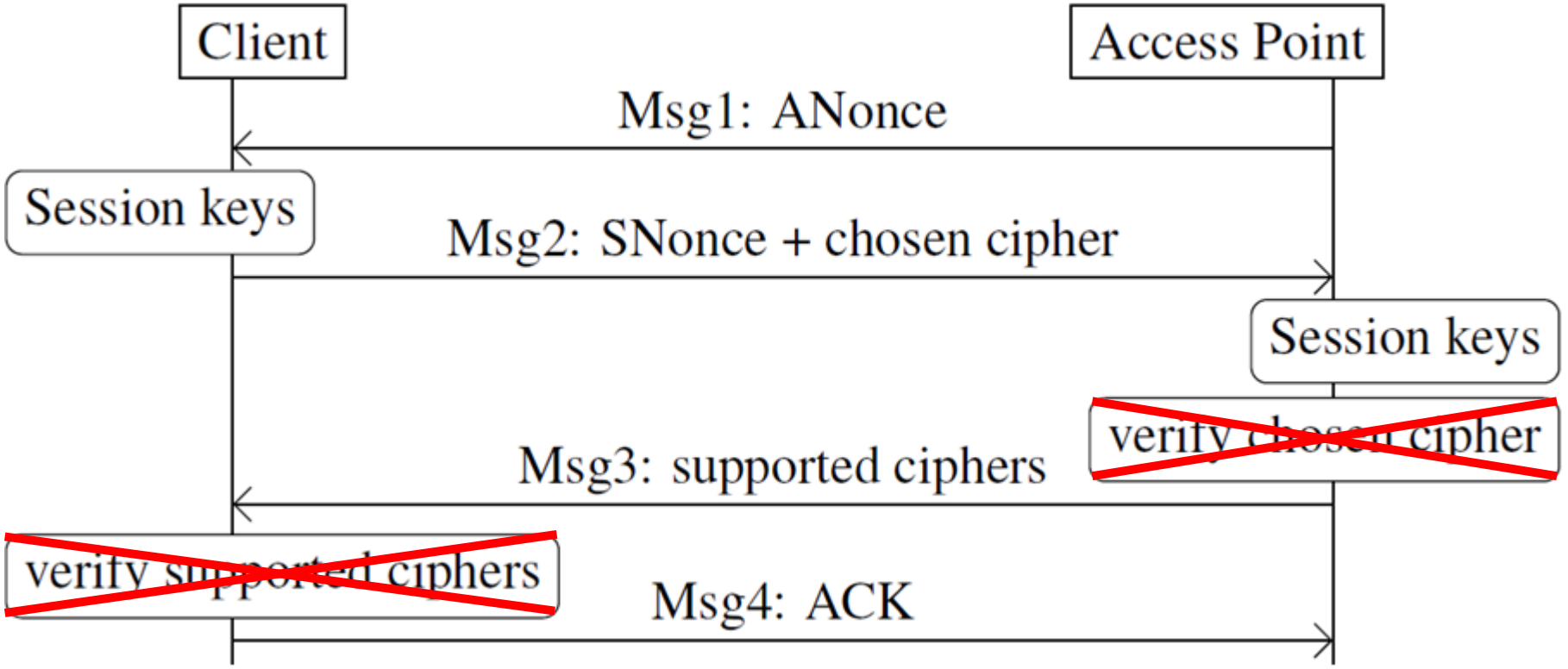
We tested 12 access points:

- Open source: OpenBSD, Linux's Hostapd
- Leaked source: Broadcom, MediaTek (home routers)
- Closed source: Windows, Apple, …
- Professional equipment: Aerohive, Aironet

Discovered several issues!

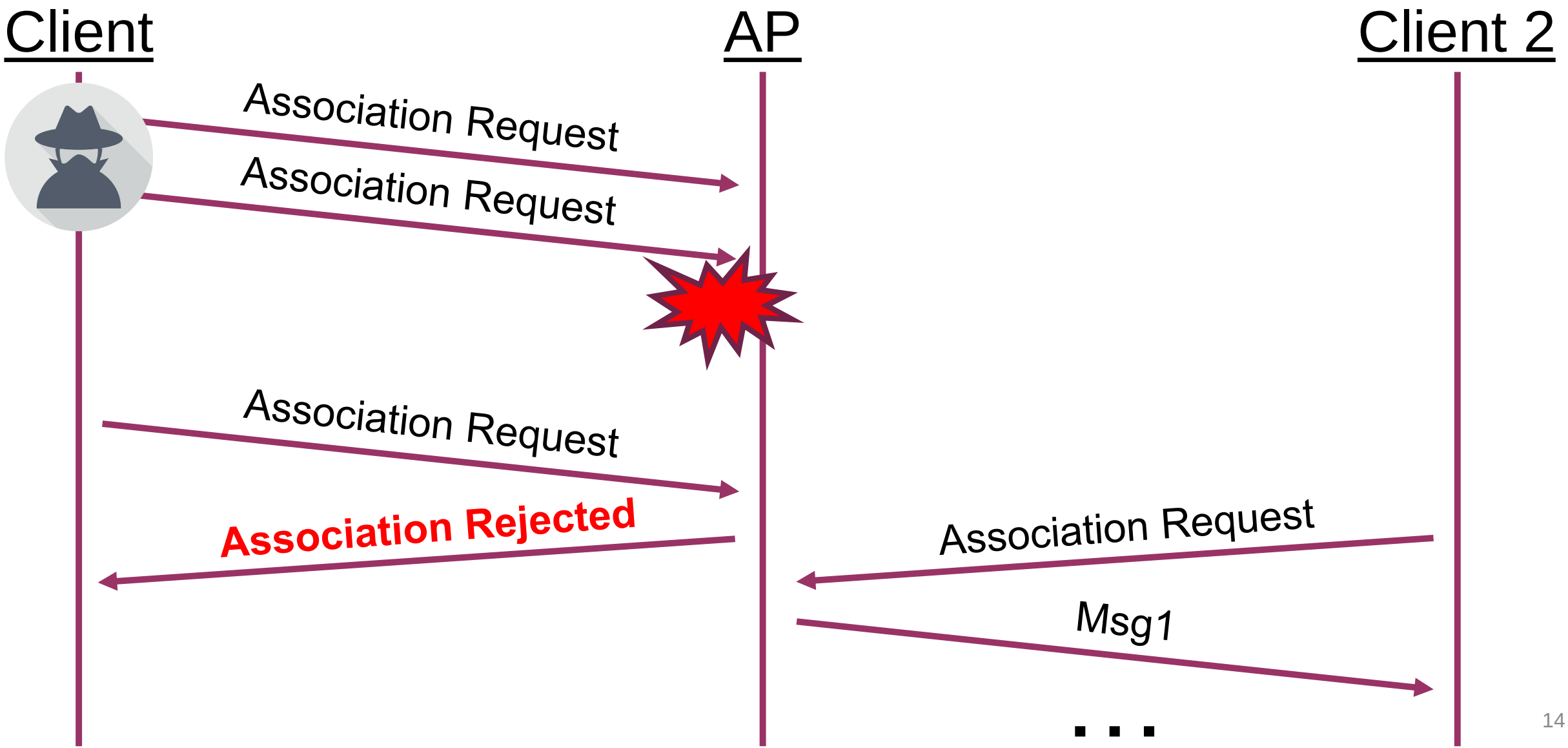
Missing downgrade checks

1. MediaTek & Telenet don't verify selected cipher in message 2
2. MediaTek also ignores supported ciphers in message 3

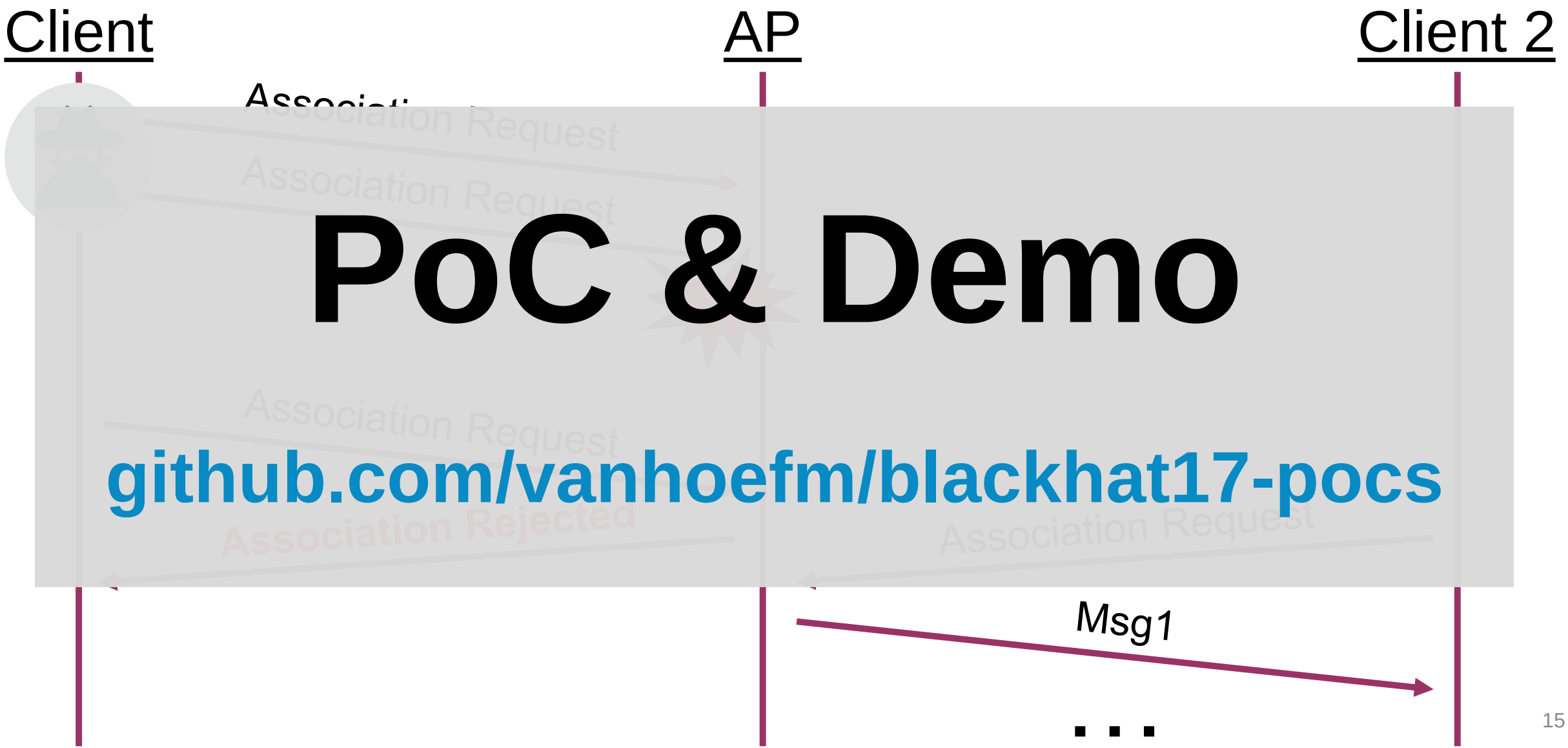


→ Trivial downgrade attack against MediaTek clients

Windows 7 targeted DoS



Windows 7 targeted DoS



PoC & Demo

github.com/vanhoefm/blackhat17-pocs

Broadcom downgrade

Broadcom cannot distinguish message 2 and 4

- Can be abused to downgrade the AP to TKIP

Hence message 4 is essential in preventing downgrade attacks

- This highlights incorrect claims in the 802.11 standard:

> "**While Message 4 serves no cryptographic purpose**, it serves as an acknowledgment to Message 3. **It is required to ensure reliability** and to inform the Authenticator that the Supplicant has installed the PTK and GTK and hence can receive encrypted frames."

OpenBSD: DoS against AP

Two bugs in OpenBSD:

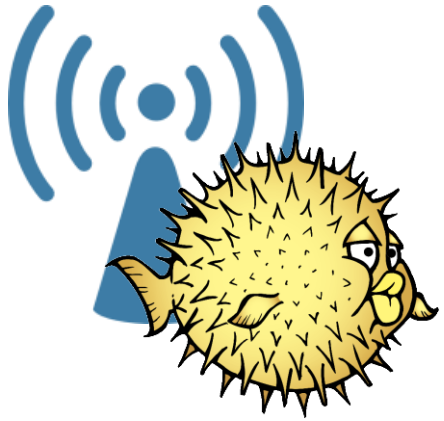
1. TKIP countermeasures are never stopped
   - Recall: it uses a weak Message Integrity Check (MIC)

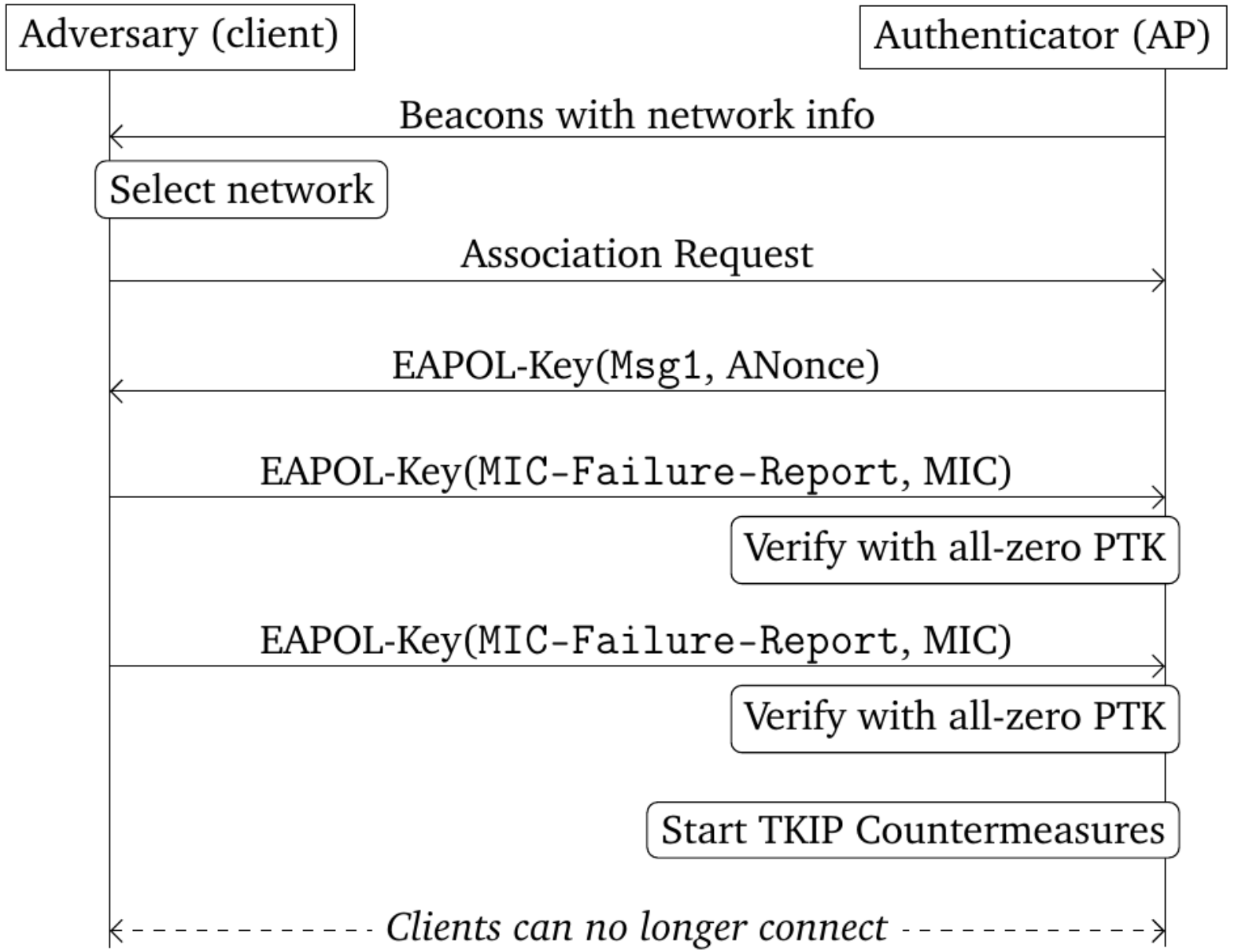
If (two MIC failures within a minute)
halt all traffic ~~for 1 minute~~ forever

2. MIC failure report accepted before 4-way handshake

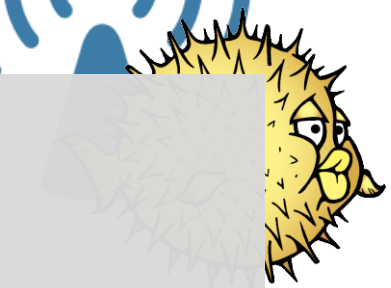
Combined: unauthenticated permanent DoS

OpenBSD: DoS against AP





OpenBSD: DoS against AP



Adversary (client)

Authenticator (AP)

Beacons with network info

Select network

Association Request

EAPOL-Key(Msg1, ANonce)

EAPOL-Key(MIC-Failure-Report, MIC)

Verify with all-zero PTK

Verify with all-zero PTK

Start TKIP Countermeasures

Clients can no longer connect

PoC & Demo

github.com/vanhoefm/blackhat17-pocs

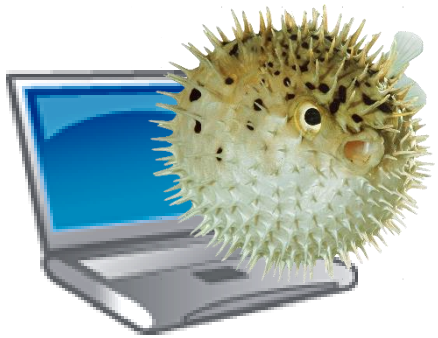
OpenBSD: client man-in-the-middle

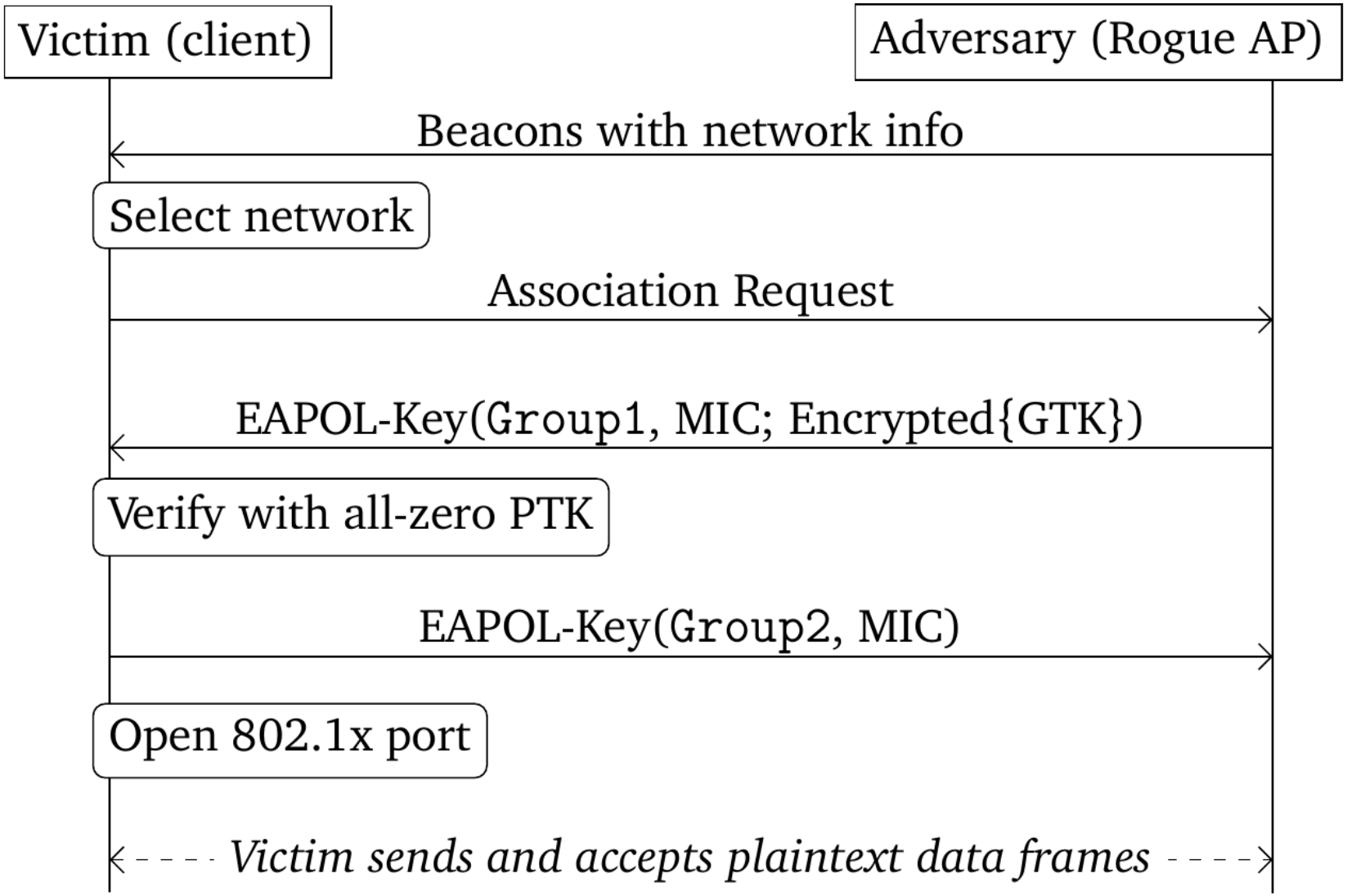
Manual inspection of OpenBSD client:

State machine missing!

→ Man-in-the-middle against client

OpenBSD: client man-in-the-middle





OpenBSD: client man-in-the-middle

PoC & Demo

github.com/vanhoefm/blackhat17-pocs

More results

See Black Hat & AsiaCCS paper[1]:

- Benign irregularities → fingerprint
- Permanent DoS attack against Broadcom
- DoS attack against Windows 10, Broadcom, Aerohive
- Inconsistent parsing of supported cipher suite list
- …

[1] M. Vanhoef, D. Shepers, and F. Piessens. Discovering Logical Vulnerabilities in the Wi-Fi Handshake Using Model-Based Testing.

Future work!

Current limitations:

- Amount of code coverage is unknown
- Only used well-formed (albeit invalid) packets
- Test generation rules applied independently
- Only tested Access Points (not clients)

But already a promising technique

✓ Black-box testing mechanism: no source code needed

✓ Fairly simple handshake, but still several logical bugs!

Conclusion

Wi-Fi code less secure than expected

- New attacks (will) keep appearing

Need better tools to detect logical flaws

- Current testing framework is basic
- Complex bugs remain undetected

Ongoing results: contact me if your product uses

- Client-side version of WPA1/2
- Other Wi-Fi handshakes: 802.11r, PeerKey, …

WiFuzz: Detecting and Exploiting Logical Flaws in the Wi-Fi Cryptographic Handshake

Mathy Vanhoef

Questions?

vanhoefm

DistriNet