Predicting and Abusing WPA2/802.11 Group Keys

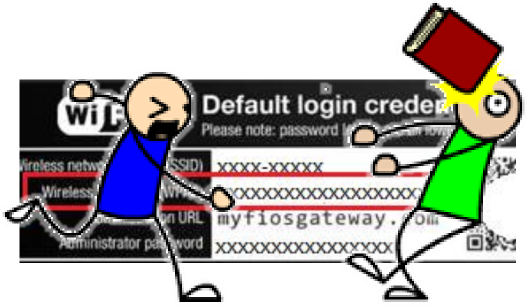
Mathy Vanhoef - imec-DistriNet, KU Leuven

@vanhoefm

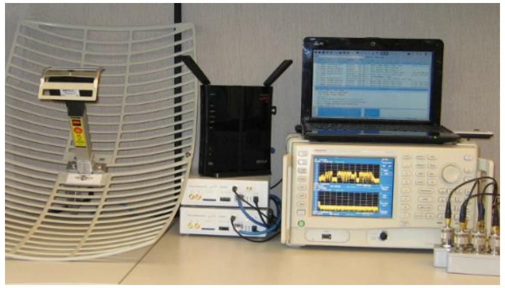
Observation

General Wi-Fi crypto is widely studied

Recover pre-shared key(s) protecting all WEP traffic


Wi Fi
Default login creden
Please note: password
Wireless netw (SSID) XXXX-XXXXX
Wireless WPA XXXXXXXXXXXXXXXXX
n URL myfiosgateway.com
Administrator password XXXXXXXXXXXXXXX

Predictable pre-shared key & dictionary attack against handshake



Rogue AP against enterprise networks to steal credentials

Tornado Attack: Recover WPA-TKIP session keys (theoretic)

→ Mainly targets pre-shared and session keys

What about group keys?

Group keys protect broadcast and multicast frames:

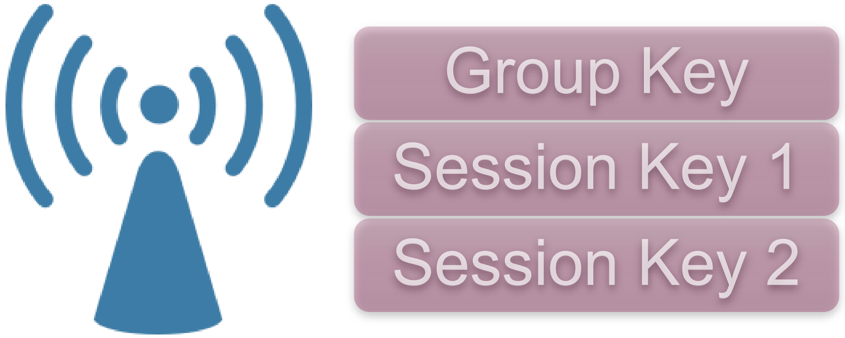
- All clients posses a copy of the group key

Security of group keys not yet properly studied!

- In contrast with pre-shared & session (=pairwise) keys …

We analyze security of group key during its full lifetime!

Background: group key lifetime



Group Key

Session Key 1

Session Key 2

Background: group key lifetime

Group Key

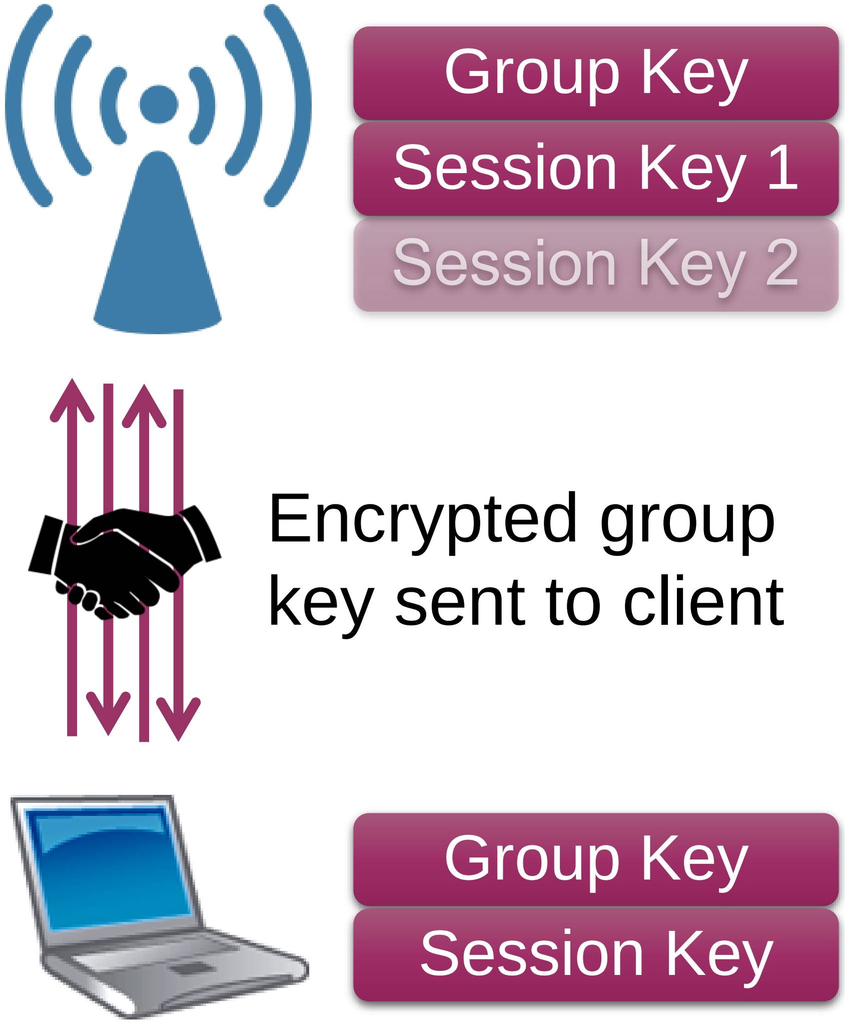
Session Key 1

Session Key 2

Three important stages:

1. Generation (flawed RNG)

Background: group key lifetime



Group Key

Session Key 1

Session Key 2

Encrypted group key sent to client

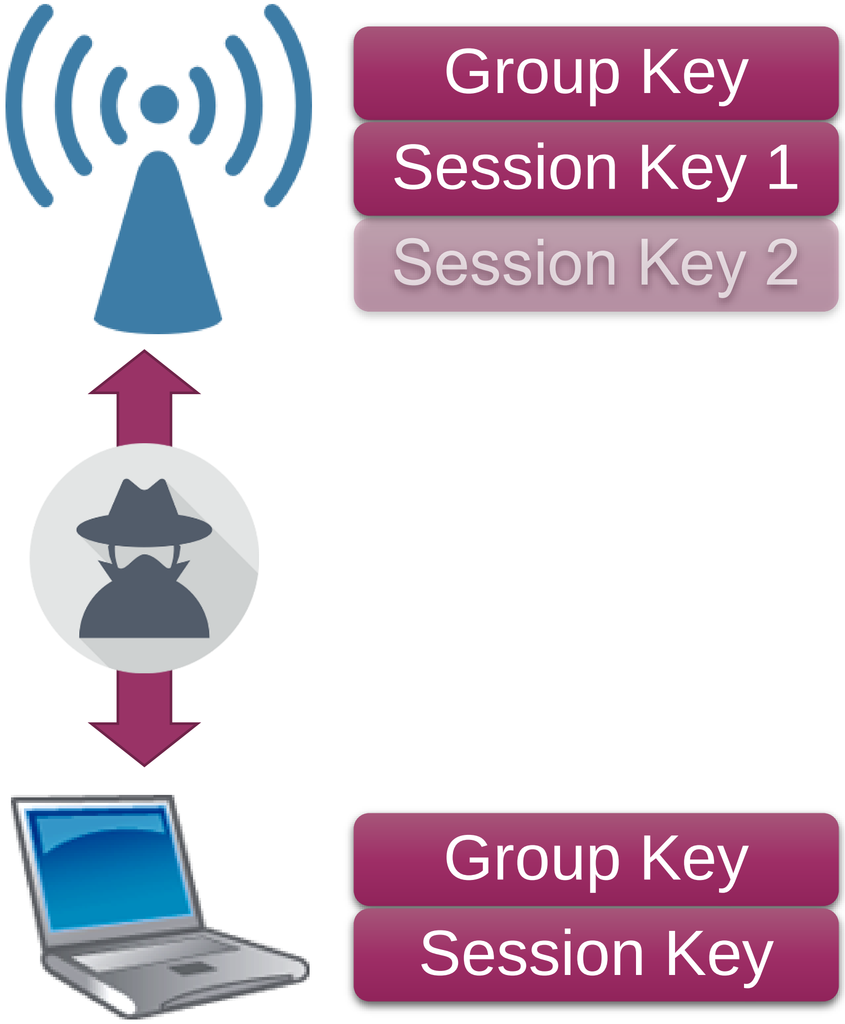
Group Key

Session Key

Three important stages:

1. Generation (flawed RNG)
2. Session key agreement and group key transport (force usage of RC4)

Background: group key lifetime



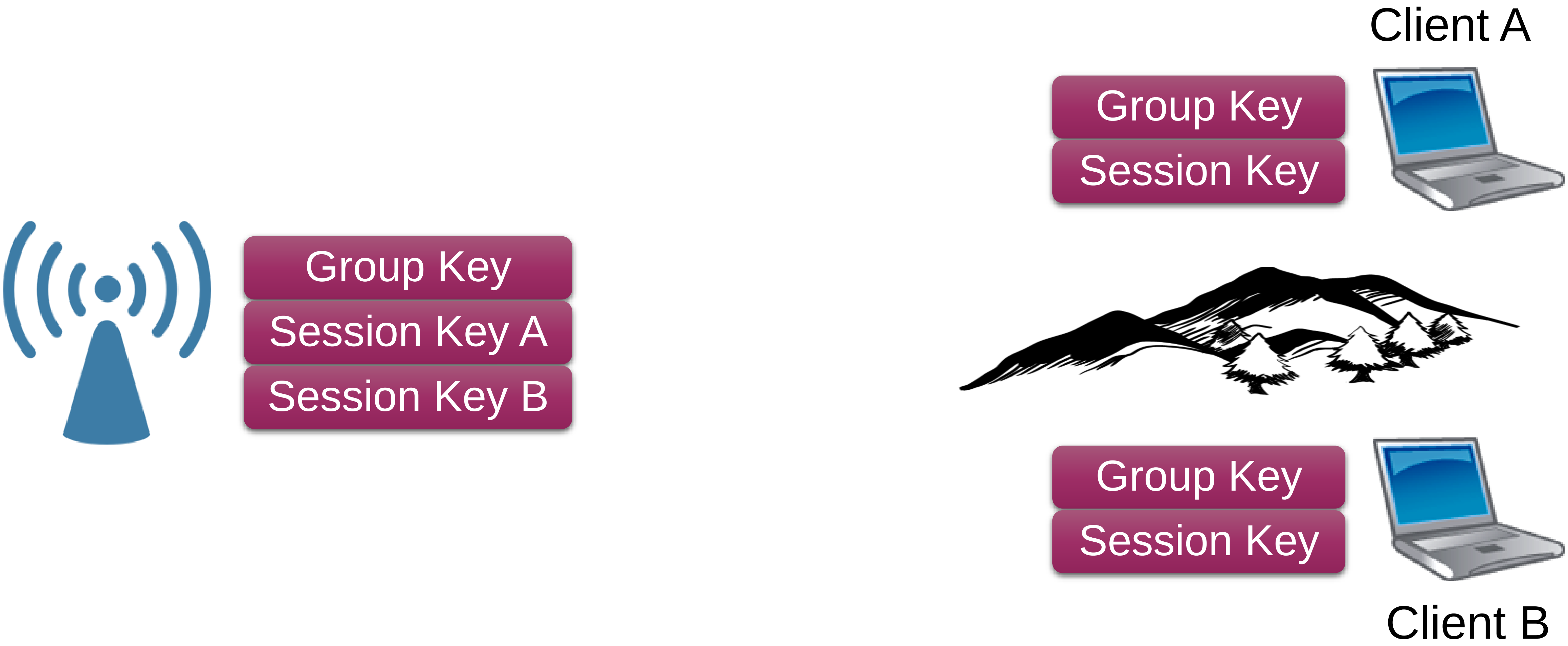
Three important stages:

1. Generation (flawed RNG)
2. Session key agreement and group key transport (force usage of RC4)
3. Usage (abuse to decrypt all traffic)

Addressing some of these issues:

- New RNG for Wi-Fi platforms?

Background: sending group frames



Client A

Group Key

Session Key

Group Key

Session Key A

Session Key B

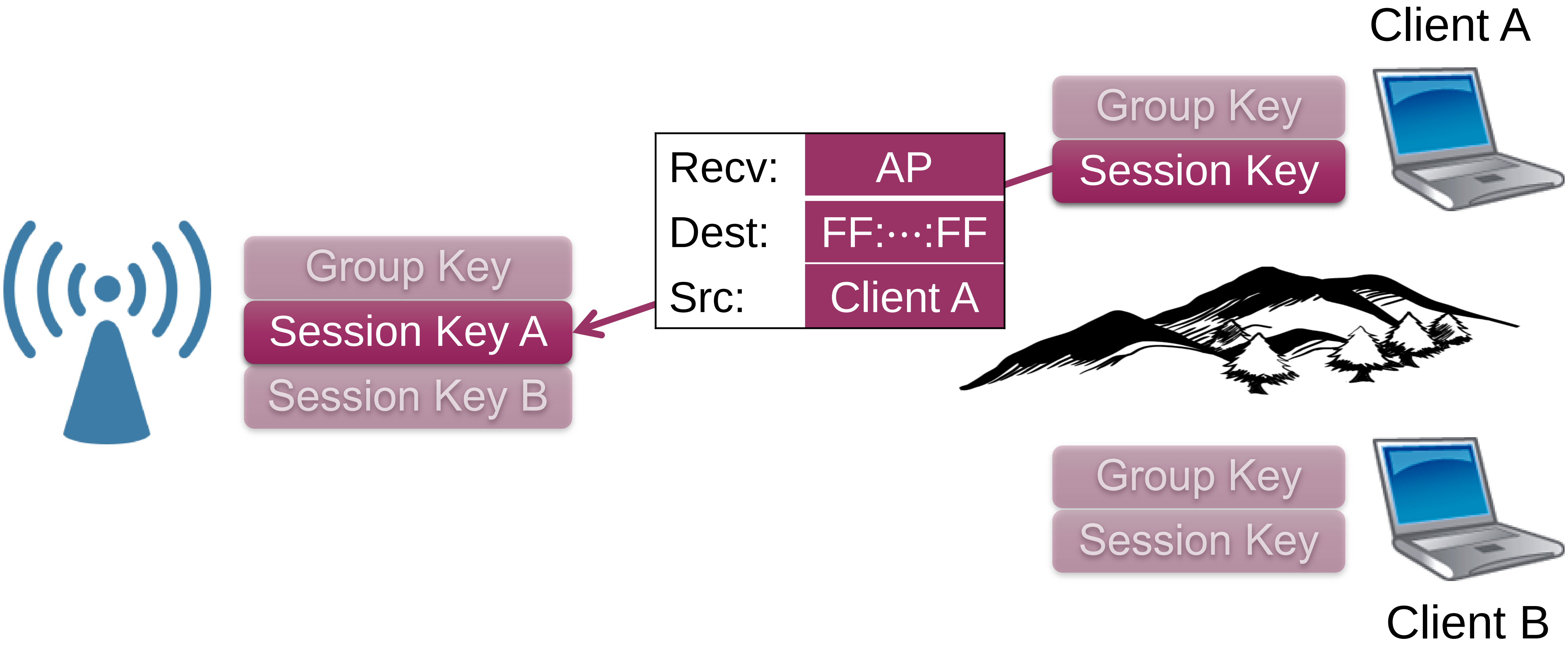
Group Key

Session Key

Client B

Background: sending group frames

1. Client uses pairwise key to send group frame to AP



Client A

Group Key

Session Key

Recv: AP

Dest: FF:⋯:FF

Src: Client A

Group Key

Session Key A

Session Key B

Group Key

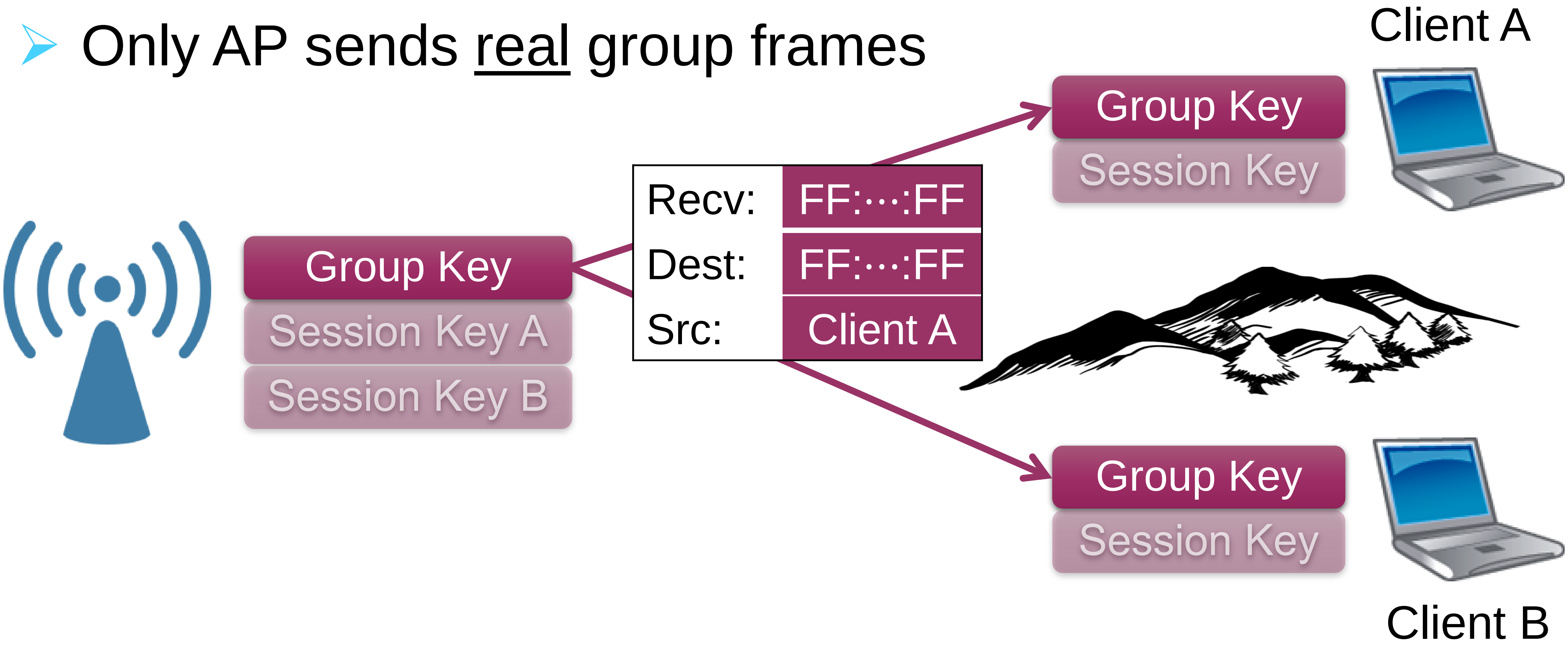
Session Key

Client B

Background: sending group frames

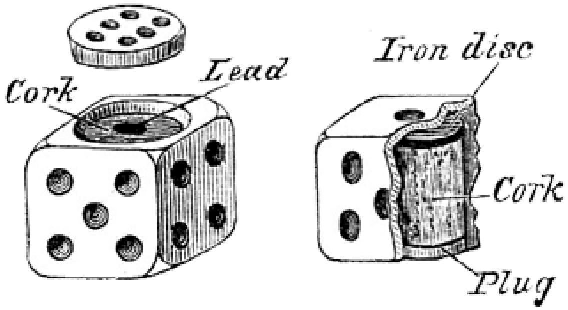
1. Client uses pairwise key to send group frame to AP
2. AP broadcasts group frame using group key

➢ Only AP sends <u>real</u> group frames



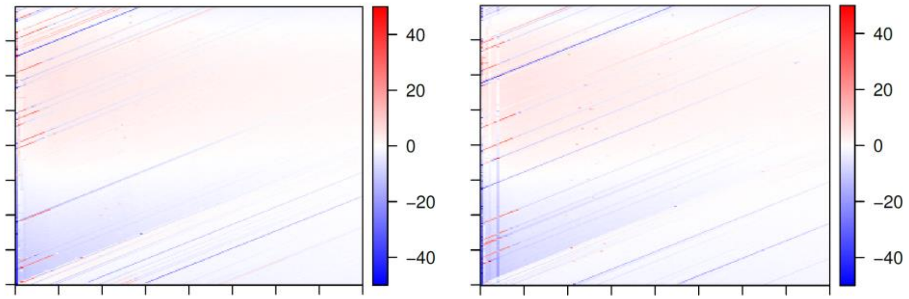
Recv:	FF:⋯:FF
Dest:	FF:⋯:FF
Src:	Client A

Agenda: security of group keys

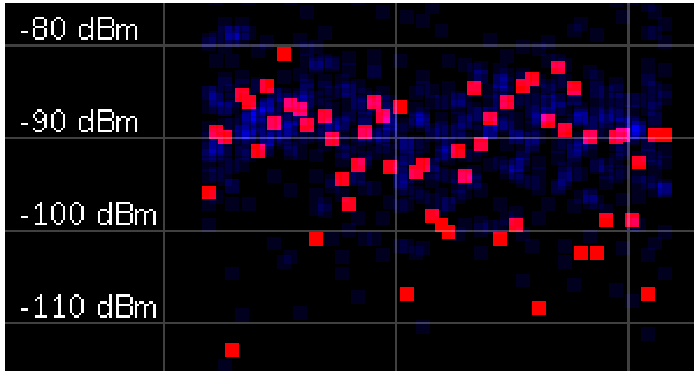


Flawed generation

Inject & decrypt all traffic

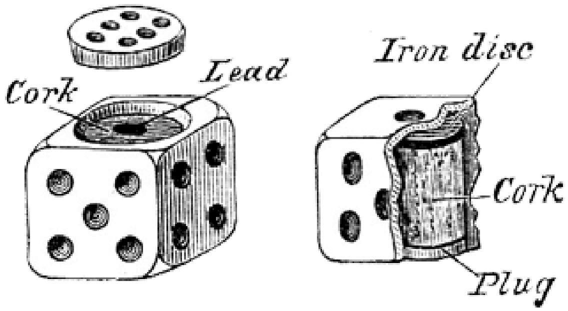


Force RC4 in handshake



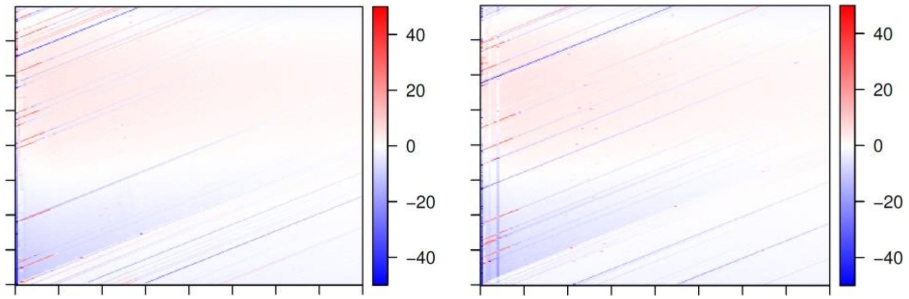
New Wi-Fi tailored RNG

Agenda: security of group keys

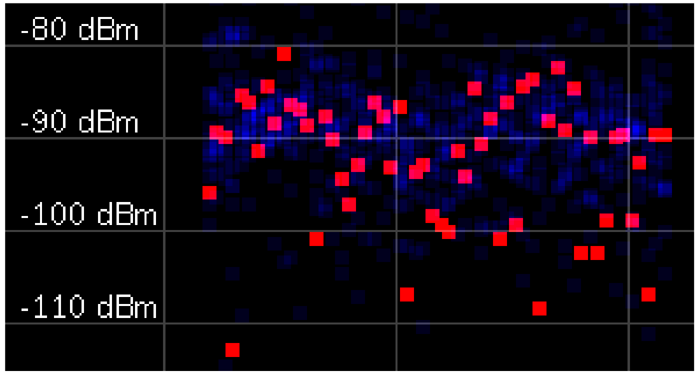


Flawed generation

Inject & decrypt all traffic



Force RC4 in handshake



New Wi-Fi tailored RNG

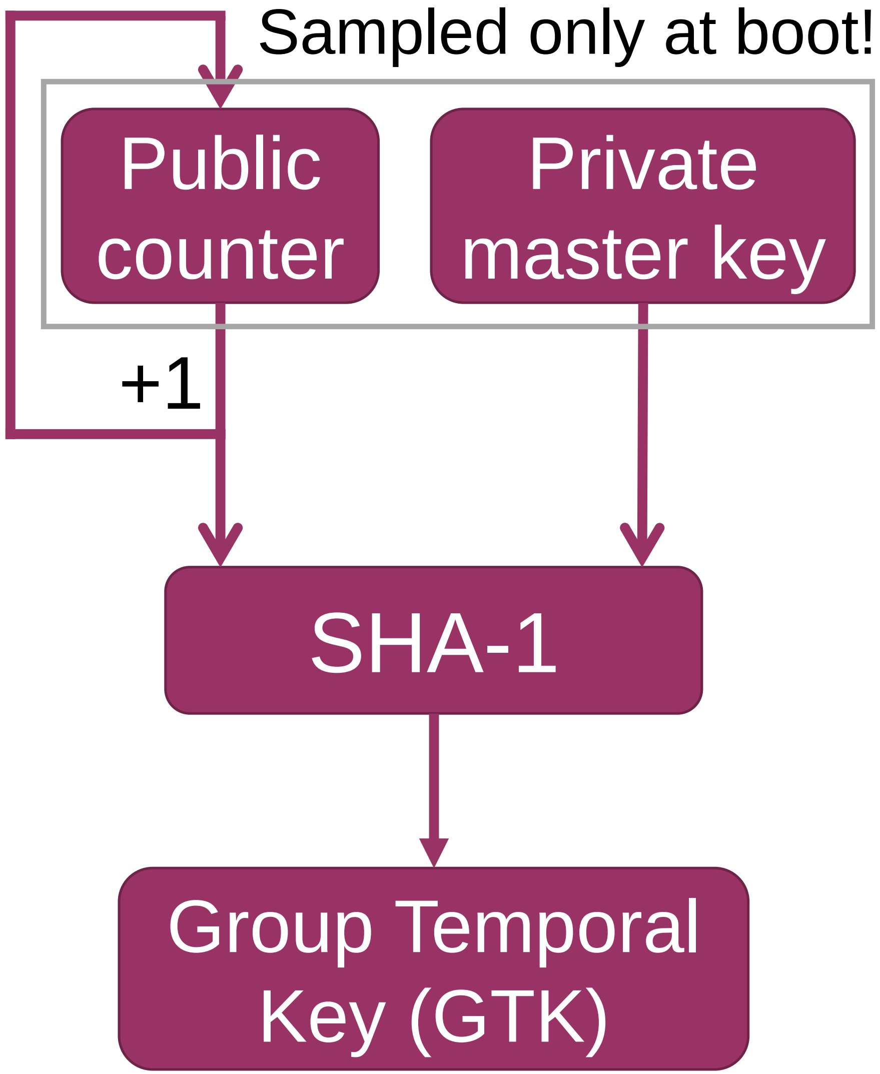
How are group keys generated?

Based on a key hierarchy:

- AP randomly generates public counter and secret master key
- Derives group temporal key (GTK) from these values every hour

Entropy only introduced at boot

- Bad design: if master key is leaked, all group keys become known!



How are random numbers generated?

802.11 standard has example Random Number Generator

- §11.1.6a: the RNG outputs cryptographic-quality randomness

> “Each STA can generate **cryptographic-quality random numbers**. This assumption is fundamental, as cryptographic methods require a source of randomness. **See M.5** for suggested hardware and software methods **to achieve randomness suitable for this purpose**.”

How are random numbers generated?

802.11 standard has example Random Number Generator

- §11.1.6a: the RNG outputs cryptographic-quality randomness
- Annex M.5: proposed RNG is expository only

“This clause suggests two sample techniques that **can be combined with the other recommendations of IETF RFC 4086** to harvest randomness. [..] These solutions are **expository only**, to demonstrate that it is feasible to harvest randomness on any IEEE 802.11 platform. [..] they do not preclude the use of other sources of randomness when available [..] ; in this case, the more the merrier. **As many sources of randomness as possible should be gathered** into a buffer, and then hashed, to obtain a seed for the PRNG.”

How are random numbers generated?

802.11 standard has example Random Number Generator

- §11.1.6a: the RNG outputs cryptographic-quality randomness
- Annex M.5: proposed RNG is expository only

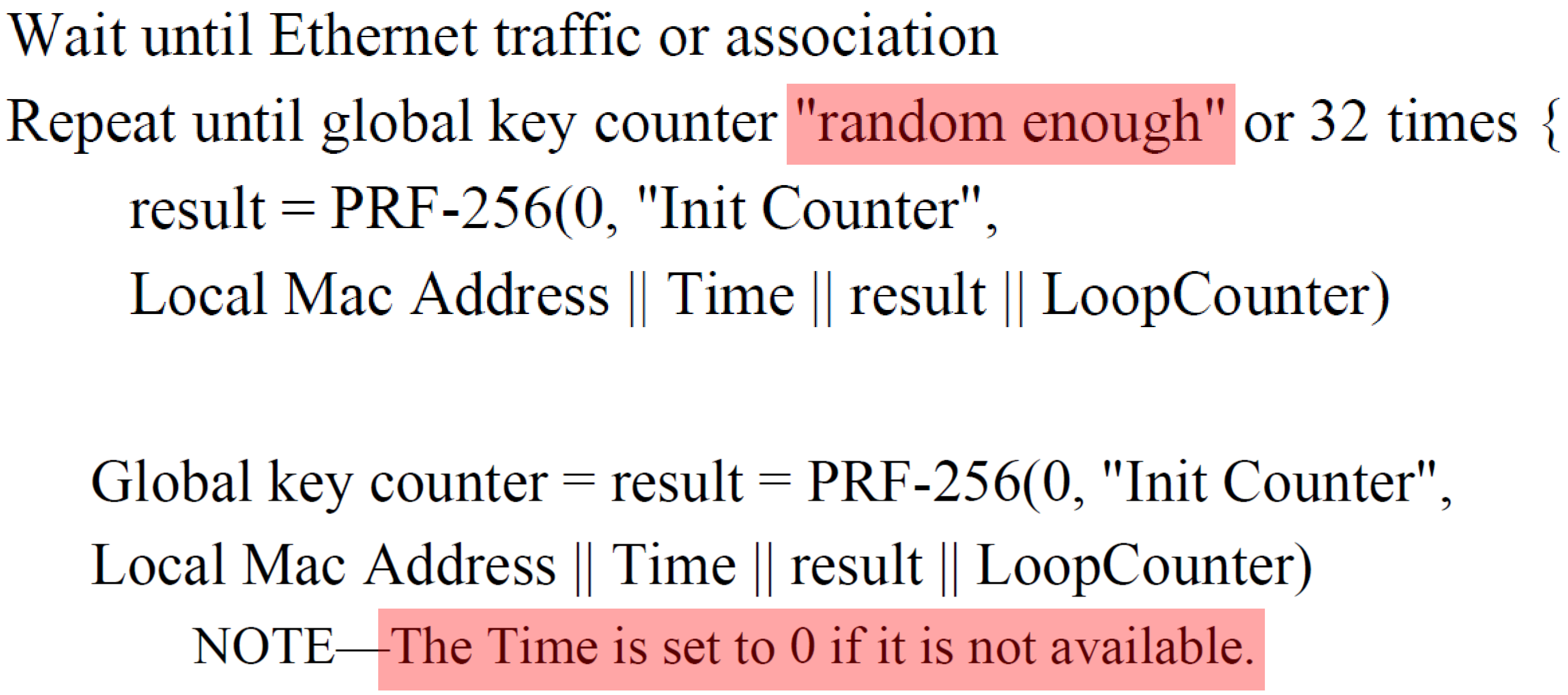
Inconsistent description of RNG’s security guarantees!

- How secure is the 802.11 RNG?
- How many platforms implement this RNG?

802.11 RNG: main design

The 802.11 RNG is a stateless function returning 32 bytes

- Vague description, even if only expository solution



```
Wait until Ethernet traffic or association
Repeat until global key counter "random enough" or 32 times {
      result = PRF-256(0, "Init Counter",
      Local Mac Address || Time || result || LoopCounter)

    Global key counter = result = PRF-256(0, "Init Counter",
    Local Mac Address || Time || result || LoopCounter)
        NOTE—The Time is set to 0 if it is not available.
```

802.11 RNG: main design

The 802.11 RNG is a stateless function returning 32 bytes

- Vague description, even if only expository solution
- Collects entropy on demand

Deviates from traditional RNG design:

- No entropy pools being maintained
- Entropy is only collected when the RNG is being invoked

802.11 RNG: main design

The 802.11 RNG is a stateless function returning 32 bytes

- Vague description, even if only expository solution
- Collects entropy on demand
- Based on frame arrival timestamps and clock jitter

802.11 RNG: entropy sources

Frame arrival times:

- Collected by starting & aborting handshakes
- Problem: AP will be blacklisted by clients

Clock jitter and drift:

- No minimum time resolution → small clock jitter
- Hence contains only low amount of randomness

¯_(ツ)_/¯

Surely no one implemented this…?

MEDIATEK

Weakened 802.11 RNG

BROADCOM

Depends on OS

Estimated ~22% of Wi-Fi networks

```
Apple PowerMac10,1 4.8.9f1 BootROM
Copyright 1994-2004 Apple Computer,
All Rights Reserved.

Welcome to                the system

To continue booting, type "mac-boot"
To shu                        and p

Release keys to continue!

 ok
0 >
```

Open Firmware

Custom RNG

Linux Hotspot

Hostapd: /dev/random

Surely no one implemented this...?

Weakened 802.11 RNG

Depends on OS

Estimated ~22% of Wi-Fi networks

```
Apple PowerMac10,1 4.8.9f1 BootROM
Copyright 1994-2004 Apple Computer,
All Rights Reserved.

Welcome to                 the system

To con        booting, type "mac-boot"
To shu                           and p

Release keys to continue!

 ok
0 >
```

Custom RNG

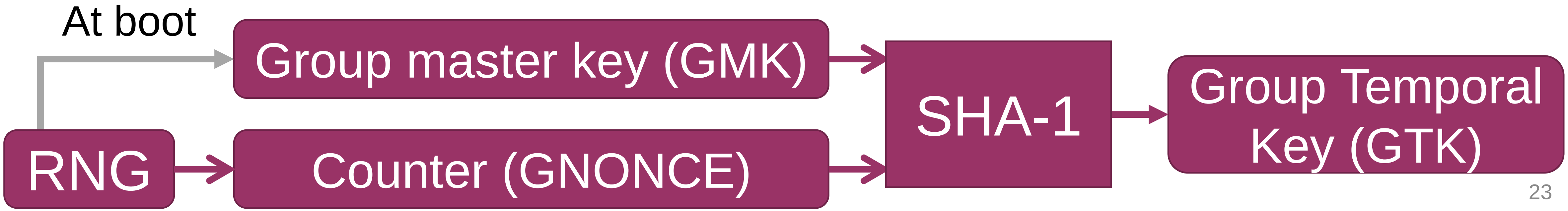
Hostapd: /dev/random

MediaTek RNG: overview

Uses custom Linux drivers:

- Implements 802.11’s group key hierarchy
  - But GNONCE “counter” is randomly refreshed on GTK rekey
- Based on the 802.11 RNG using **only clock jitter**
- Uses *jiffies* for current time: equals uptime of the AP

➢ Predict both GMK and GNONCE to determine group key!



MediaTek RNG: key search

- Jiffies have at best millisecond accuracy
- GMK: generated at boot → limited set of possible values
- GNONCE: depends on uptime of router (and clock skew)
  - Uptime is leaked in beacons

➢ Capture encrypted broadcast packet and search for key ☺

RT-AC51U

OpenCL

~3 mins

GMK & GTK

MediaTek: predicting the GTK

DEMO

Surely no one implemented this...?

MEDIATEK

Weakened 802.11 RNG

BROADCOM

Depends on OS

Estimated ~22% of Wi-Fi networks

```
Apple PowerMac10,1 4.8.9f1 BootROM
Copyright 1994-2004 Apple Computer,
All Rights Reserved.

Welcome to                          the system

To continue booting, type "mac-boot"
To shu                              and p

Release keys to continue!

 ok
0 >
```

Open Firmware

Custom RNG

Linux Hotspot

Hostapd: /dev/random

Broadcom: Linux

When running on a Linux kernel:

- Implements 802.11’s group key hierarchy
- Randomness from `/dev/urandom`

“Mining your Ps and Qs” by Heninger et al.:

- `/dev/urandom` might be predictable at boot
- All group keys might be predictable on old kernels

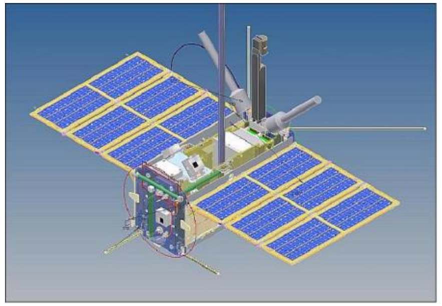
Broadcom: VxWorks and eCos

VxWorks

Proprietary

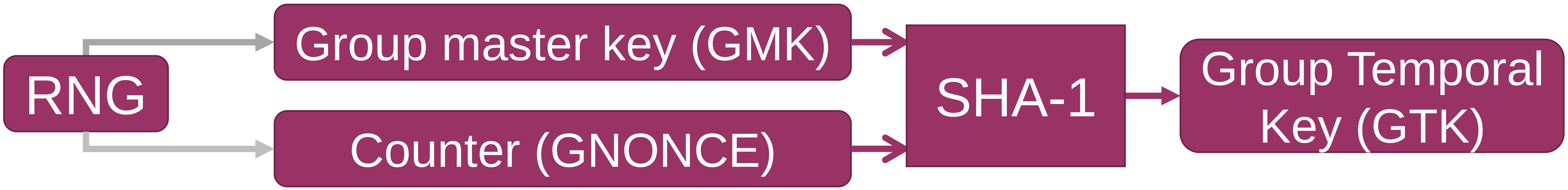
eCos

Open Source



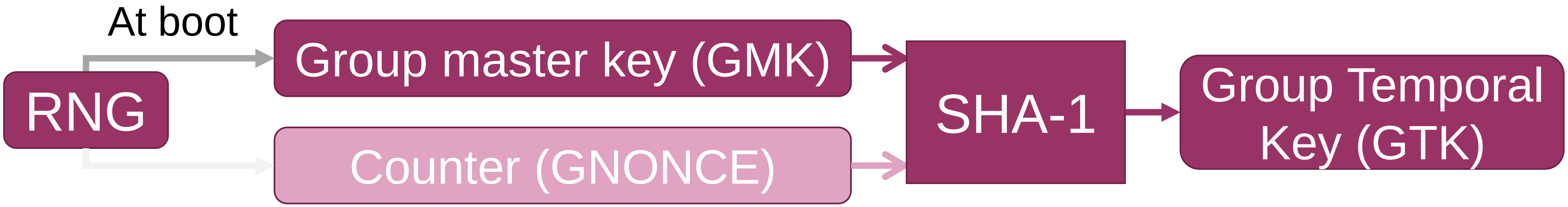
Broadcom: VxWorks and eCos

- Implements 802.11's group key hierarchy
- Random numbers: MD5(time in microseconds)



Broadcom: VxWorks and eCos

- Implements 802.11's group key hierarchy
- Random numbers: MD5(time in microseconds)
- GNONCE counter is leaked during handshake
- ➢ Attacker only has to predict master group key (GMK)



Broadcom: VxWorks and eCos

- Implements 802.11’s group key hierarchy
- Random numbers: MD5(time in microseconds)
- GNONCE counter is leaked during handshake

➢ Attacker only has to predict master group key (GMK)

WRT54Gv5

OpenCL

~4 mins

GMK & GTK

Surely no one implemented this…?

MEDIATEK

Weakened 802.11 RNG

BROADCOM

Depends on OS

Estimated ~22% of Wi-Fi networks

```
Apple PowerMac10,1 4.8.9f1 BootROM
Copyright 1994-2004 Apple Computer,
All Rights Reserved.

Welcome to                the system

To continue booting, type "mac-boot"
To shu                        and p

Release keys to continue!

 ok
0 >
```

Open Firmware

Custom RNG

Linux
Hotspot

Hostapd: /dev/random

Open Firmware

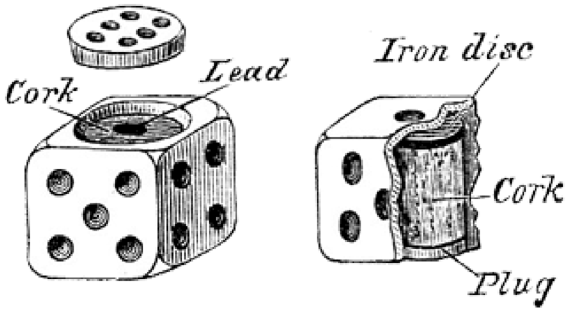
Open Firmware:

- An open source BIOS
- Supports client Wi-Fi functionality in BIOS (!)
- Randomness from boot time & linear congruential generator

Hostapd:

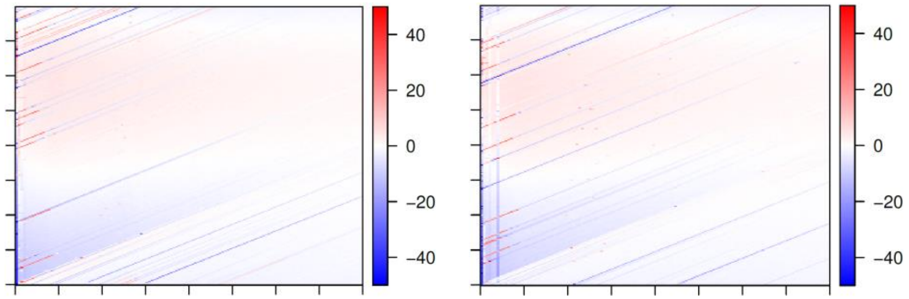
- Based on 802.11 group key hierarchy
  - Also injects new entropy on group rekeys!
- Reads from /dev/random on boot & when clients join
- If not enough entropy available, connections are rejected

Agenda: security of group keys

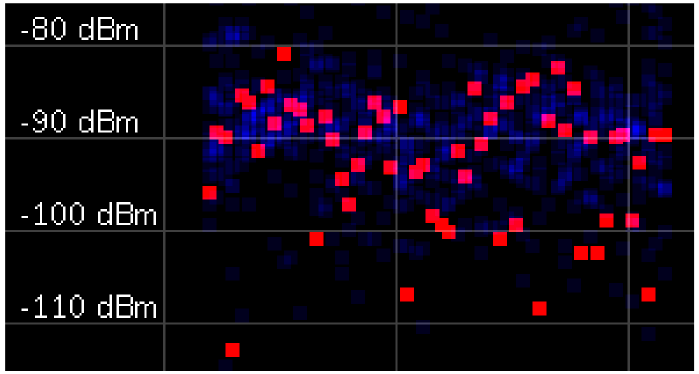


Flawed generation

Inject & decrypt all traffic



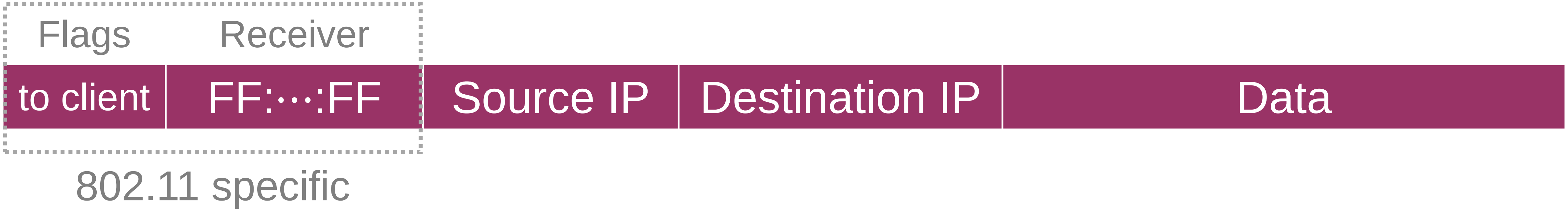
Force RC4 in handshake



New Wi-Fi tailored RNG

Injecting unicast packets?

- Put unicast IP packet in a broadcast frame?



Flags

Receiver

to client

FF:⋯:FF

Source IP

Destination IP

Data

802.11 specific

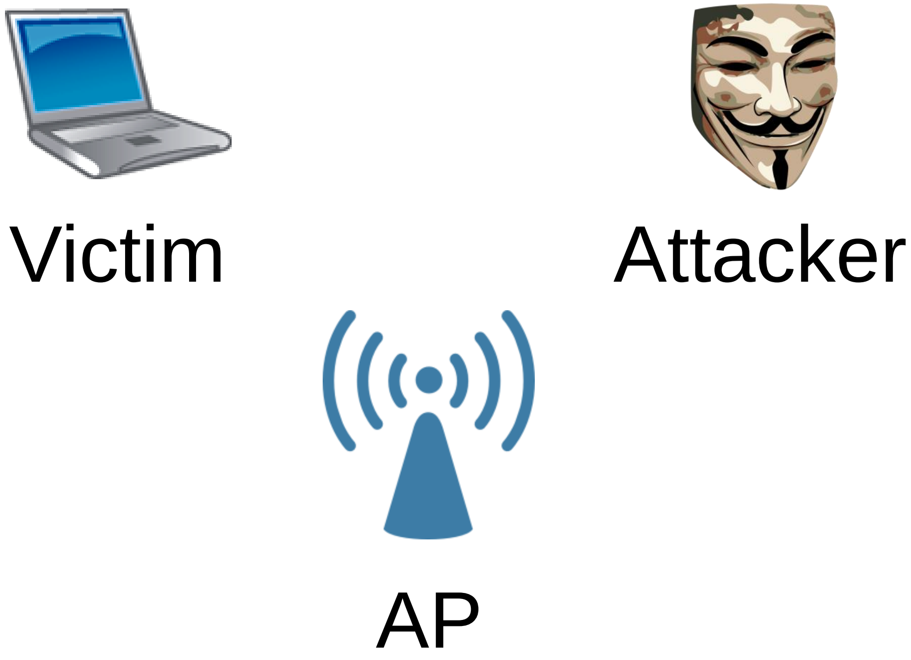
- Detected by “Hole 196” check

Hole 196 check done at network-layer …

… but an AP works at link-layer!

Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:

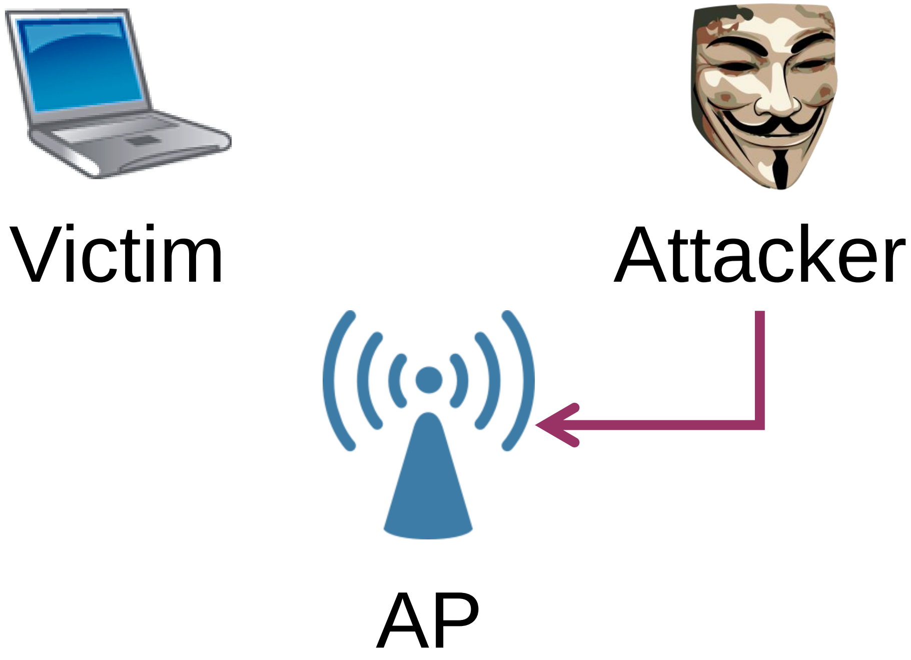


Sender	Destination	Data

Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:

1. Inject as group frame to AP



Flags	Receiver	Final dest.			
To AP	FF:⋯:FF	Victim	Sender	Destination	Data

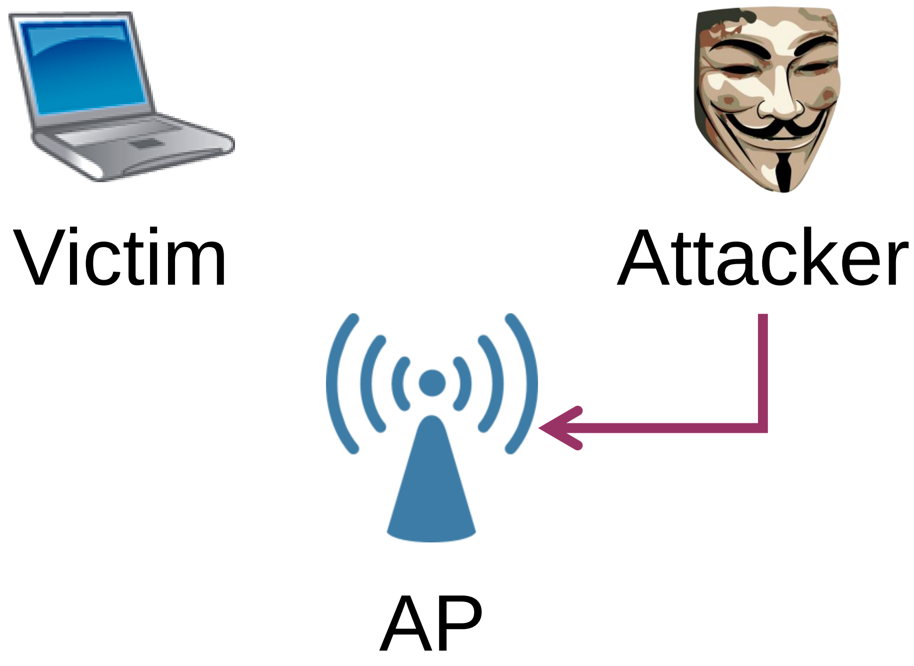
802.11 specific

Encrypted using group key

Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:

1. Inject as group frame to AP
2. AP processes and routes frame



Flags	Receiver	Final dest.			
To AP	FF:⋯:FF	Victim	Sender	Destination	Data

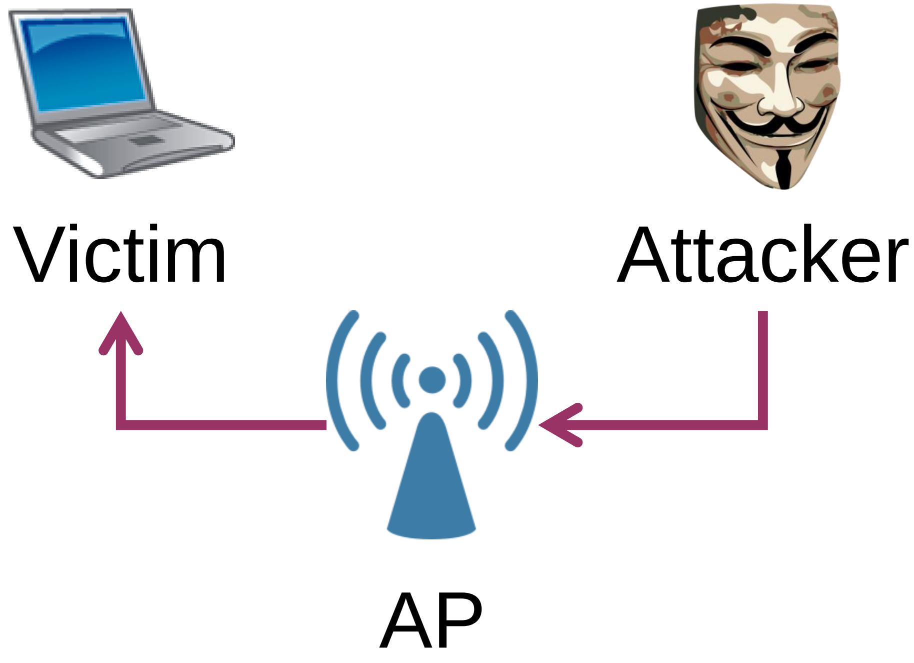
802.11 specific

Decrypted using **group key**

Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:

1. Inject as group frame to AP
2. AP processes and routes frame
3. AP transmits it to destination



Flags	Receiver	Final dest.			
To STA	Victim	Victim	Sender	Destination	Data

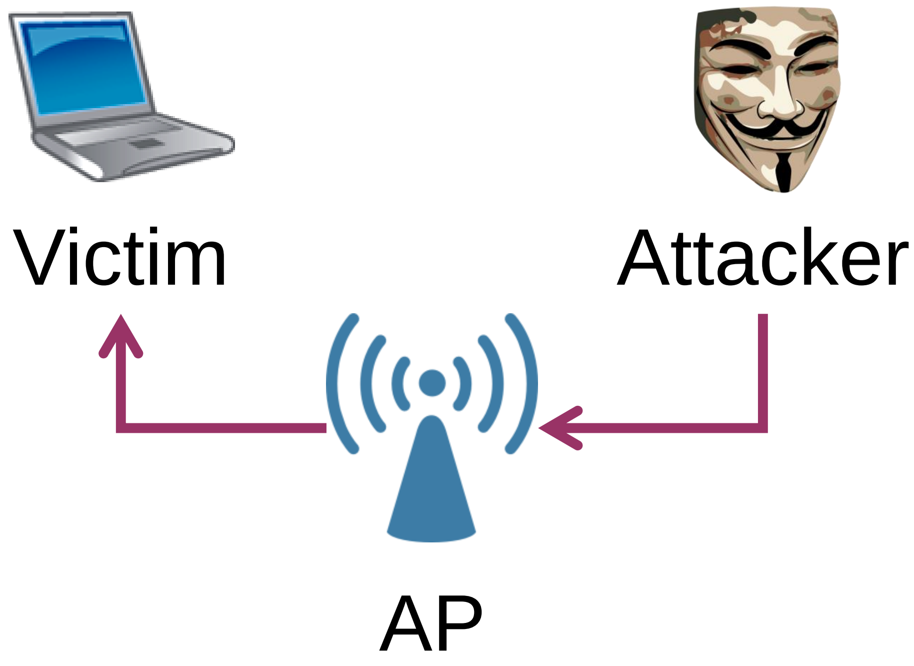
802.11 specific

Encrypted using session (pairwise) key

Forging unicast frames using group key

Abuse AP to bypass Hole 196 check:

1. Inject as group frame to AP
2. AP processes and routes frame
3. AP transmits it to destination
4. Victim sees normal unicast frame



Flags	Receiver	Final dest.			
To STA	Victim	Victim	Sender	Destination	Data

802.11 specific

Decrypted using **session (pairwise) key**

Decrypting all traffic

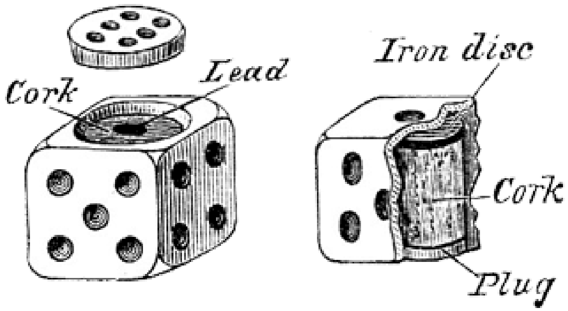
ARP poison to broadcast MAC address

- Poison both router and clients
- Can decrypt network-layer protocols: IPv4, IPv6, …

Countermeasure:

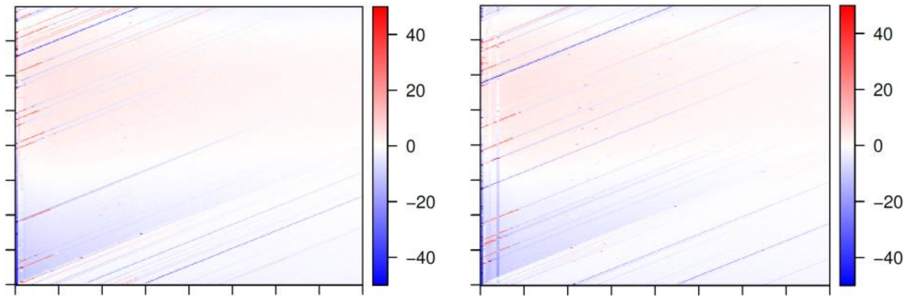
- Don’t forward broadcast frames to a unicast destination
- Even better: AP should simply ignore frames received on broadcast or multicast MAC address.

Agenda: security of group keys

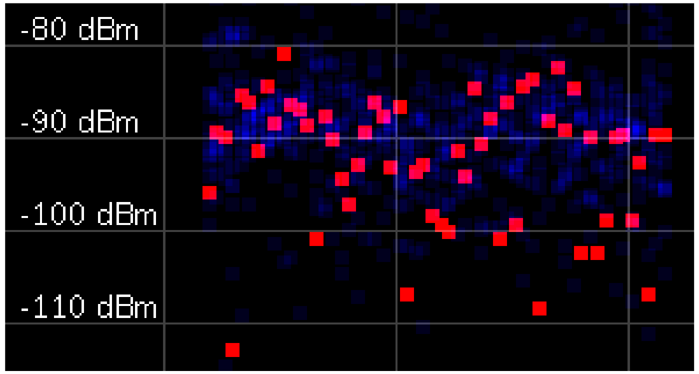


Flawed generation

Inject & decrypt all traffic

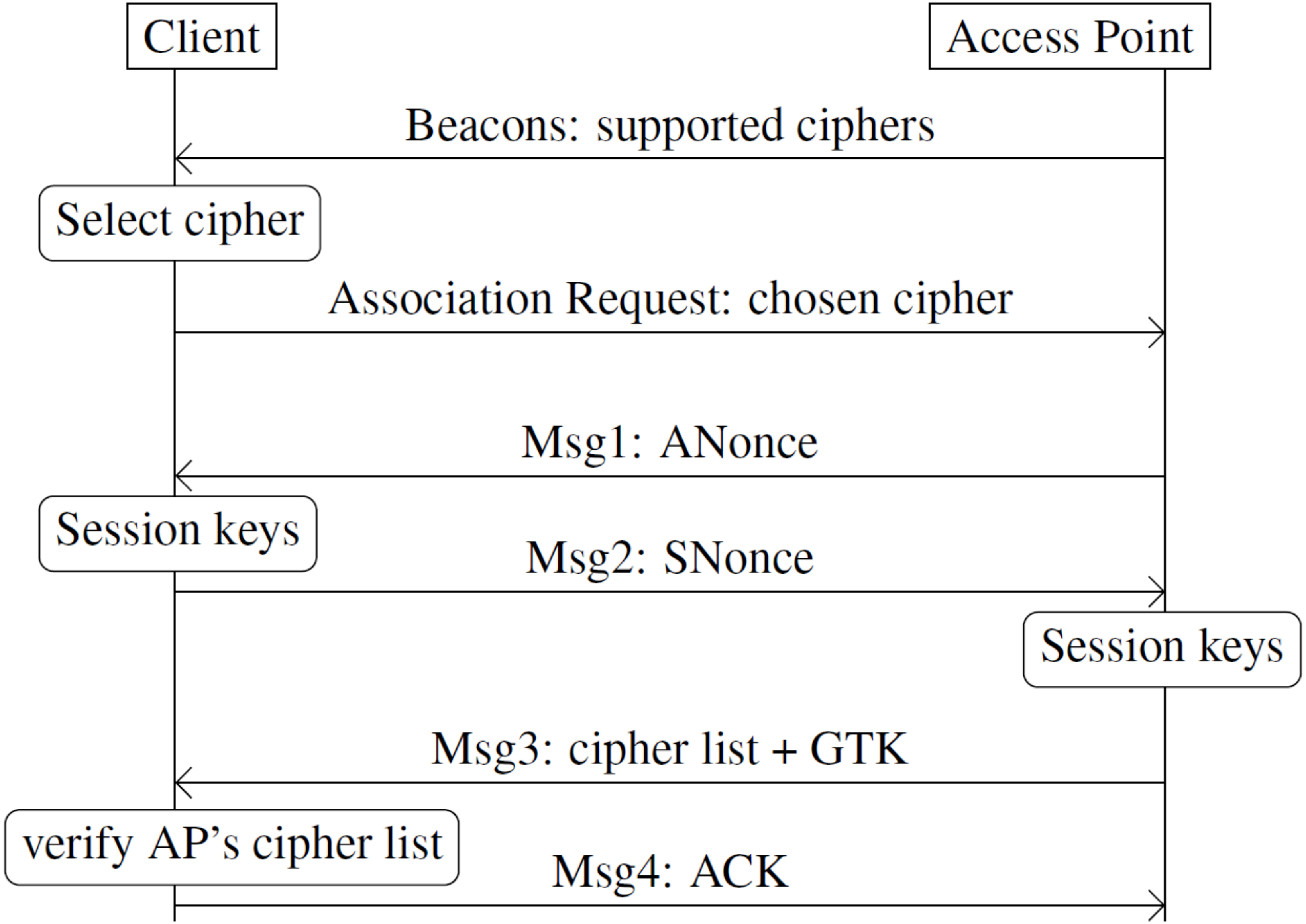


Force RC4 in handshake

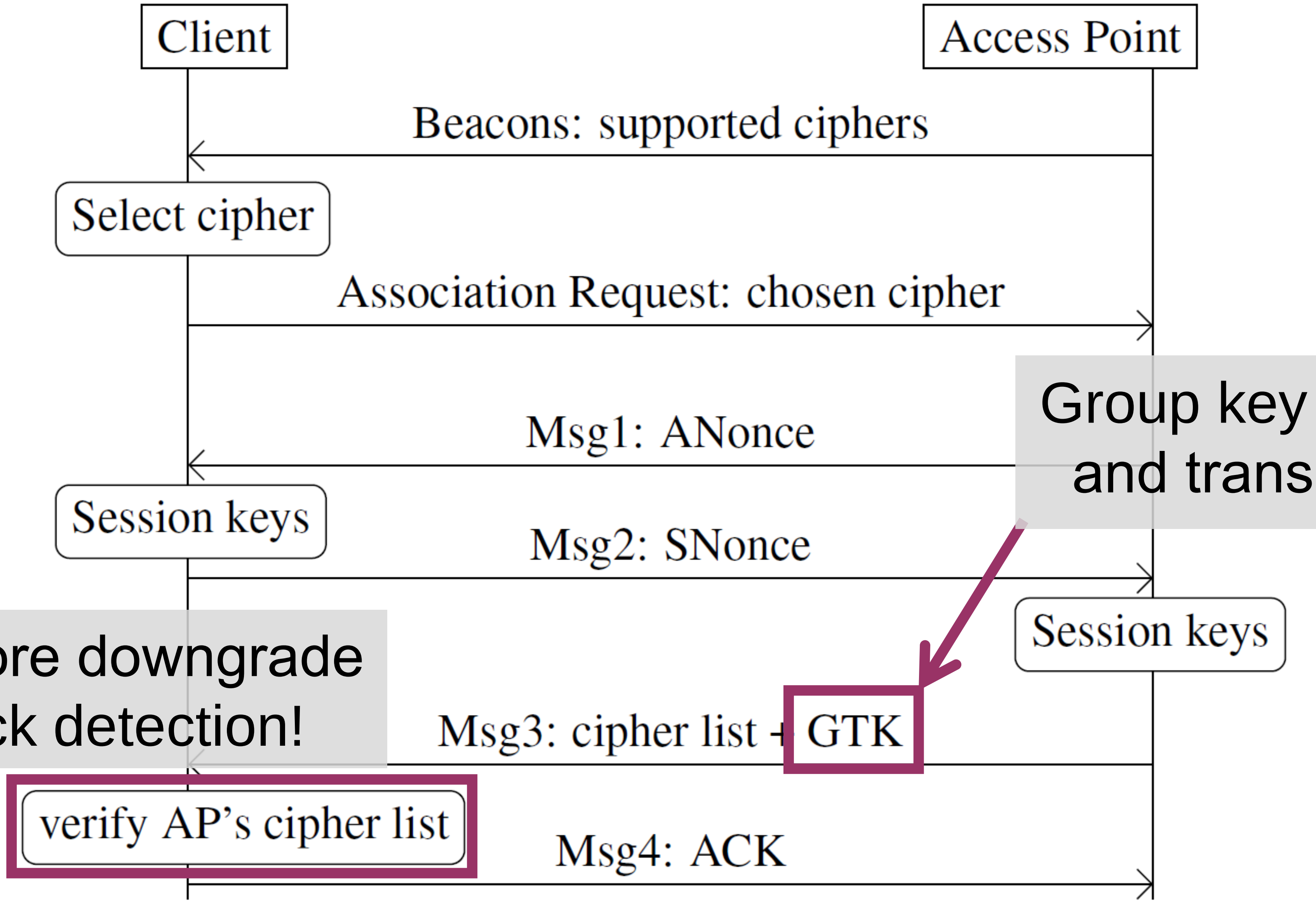


New Wi-Fi tailored RNG

The 4-way handshake



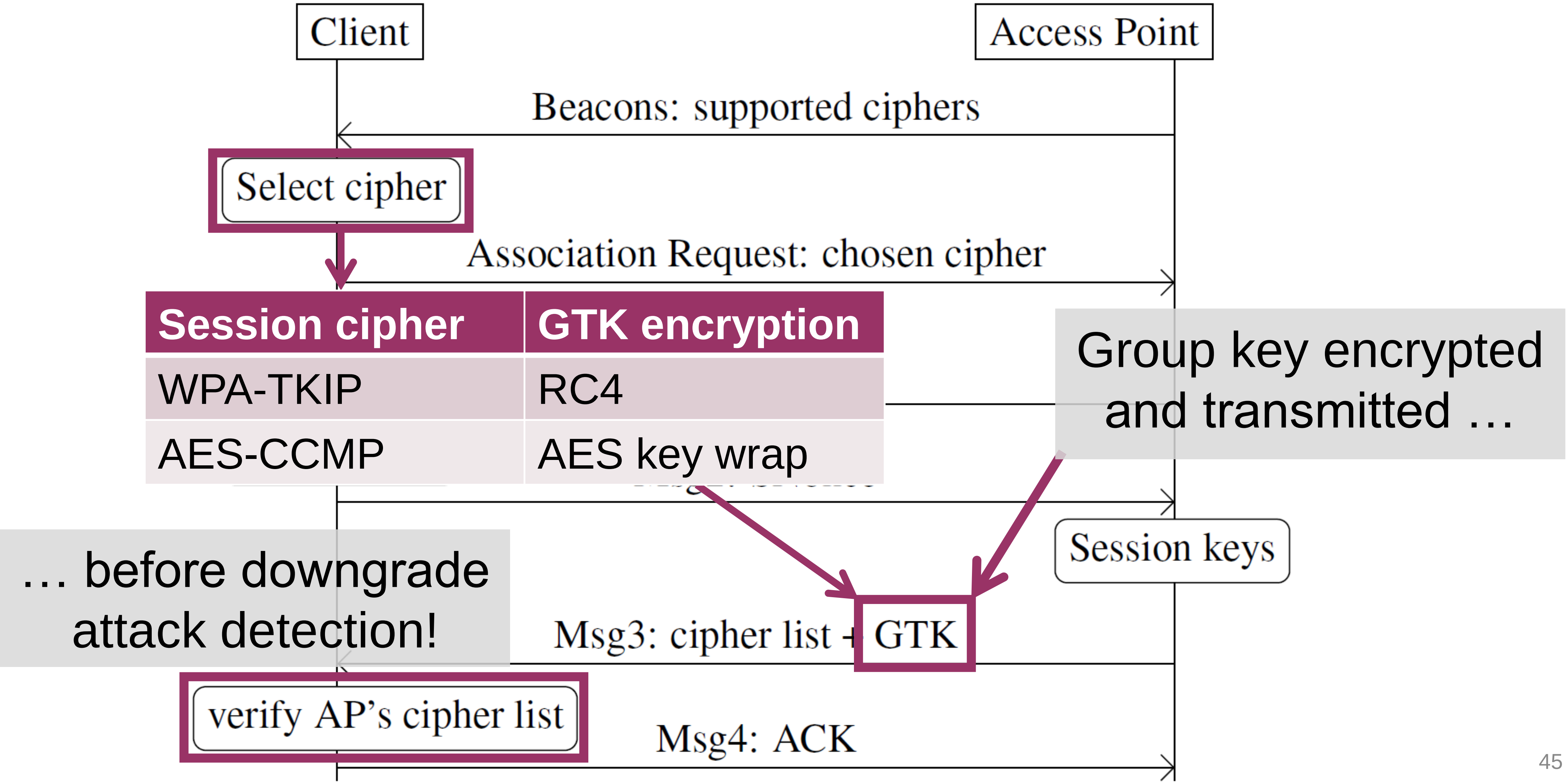
The 4-way handshake



Group key encrypted and transmitted …

… before downgrade attack detection!

The 4-way handshake



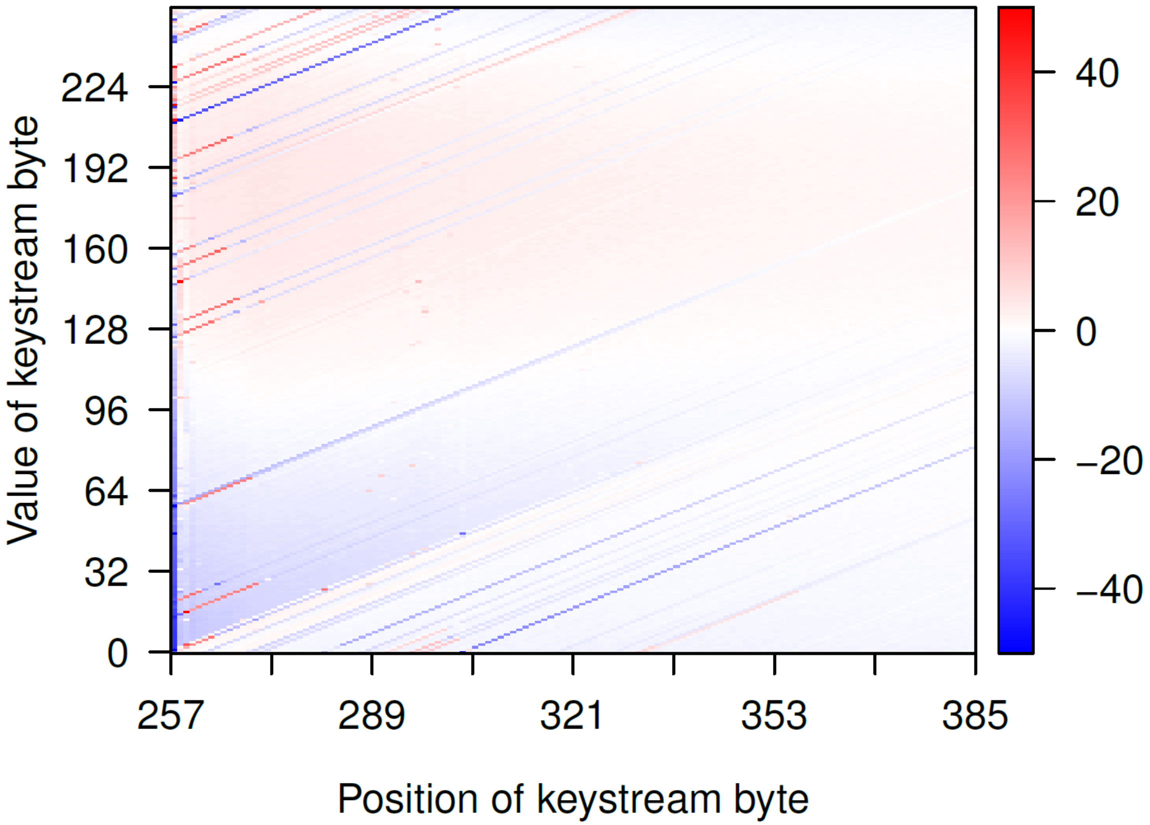
Session cipher	GTK encryption
WPA-TKIP	RC4
AES-CCMP	AES key wrap

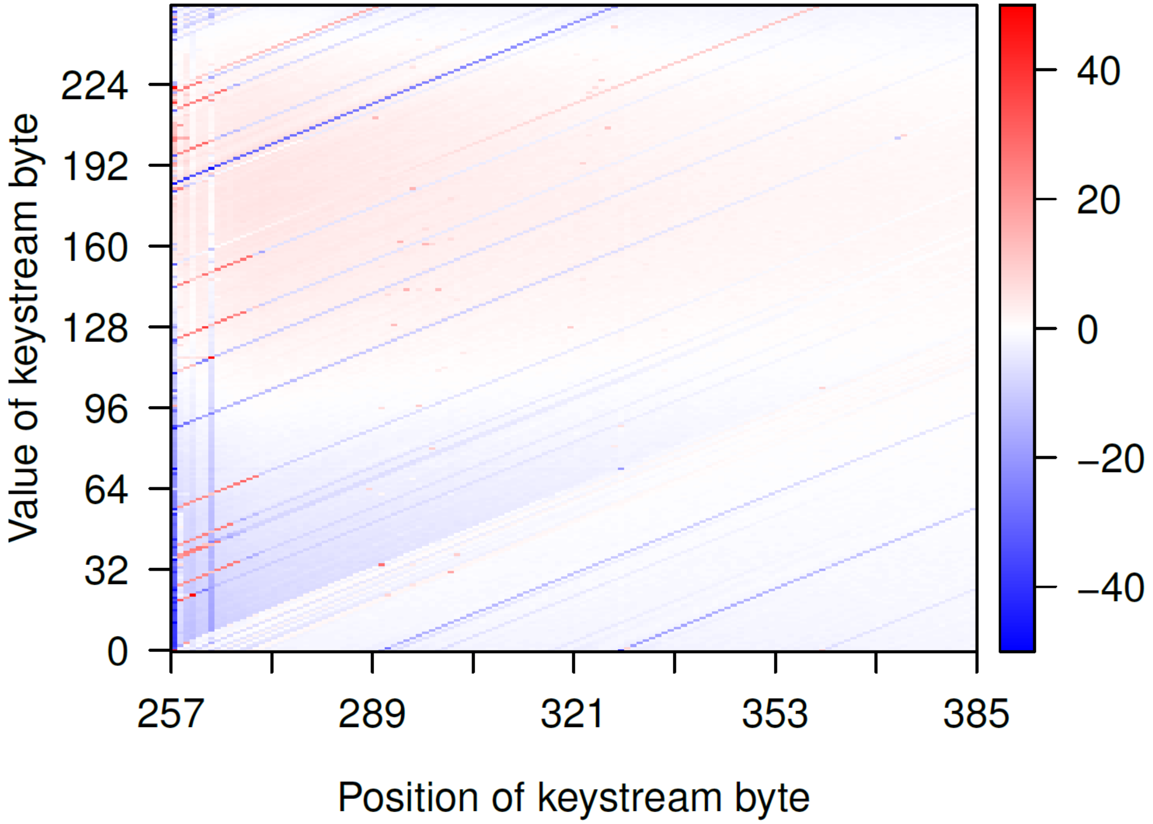
Group key encrypted and transmitted …

… before downgrade attack detection!

Attacking RC4 encryption of GTK

- RC4 Key: 16-byte IV ||16-byte secret key
- First 256 keystream bytes are dropped





Attacking RC4 encryption of GTK

- RC4 Key: 16-byte IV ||16-byte secret key
- First 256 keystream bytes are dropped

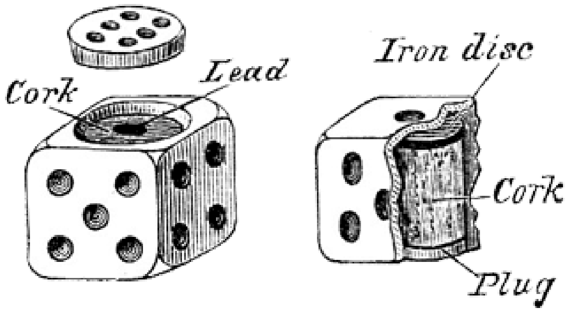
Recover repeated encryptions of GTK:

- Similar in spirit to RC4 NOMORE attack
- Requires $\sim 2^{31}$ handshakes: takes >50 years

Countermeasures:

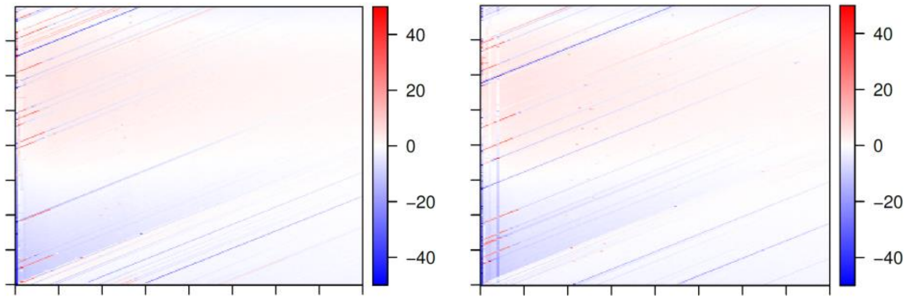
- Disable WPA-TKIP & RC4
- Send GTK after handshake

Agenda: security of group keys

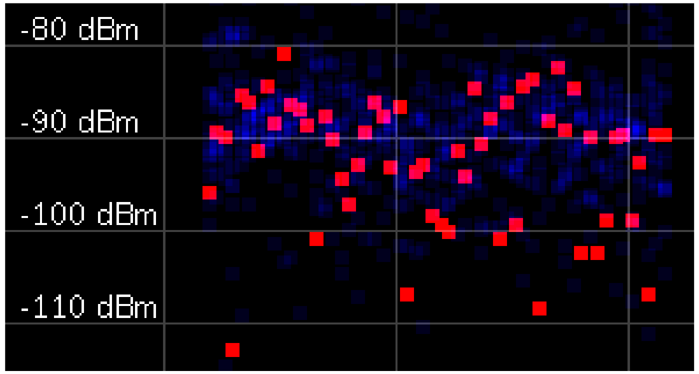


Flawed generation

Inject & decrypt all traffic

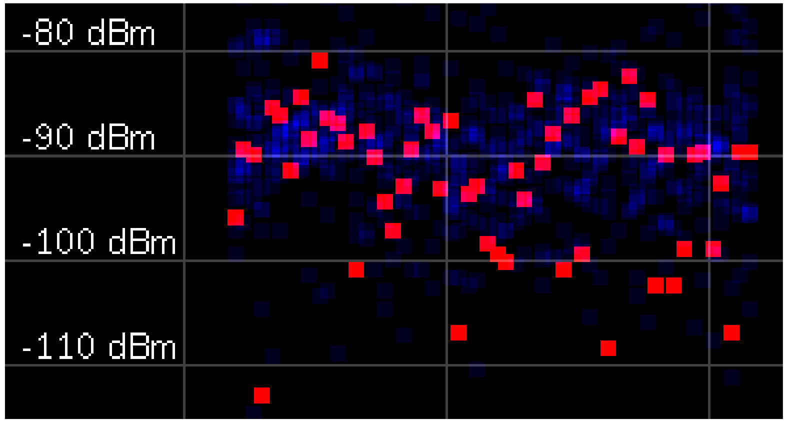


Force RC4 in handshake



New Wi-Fi tailored RNG

An improved 802.11 RNG



Entropy present on al Wi-Fi chips?

- Wi-Fi signals & background noise

Spectral scan feature in commodity chips:

- Can generate 3 million samples / second
- First XOR samples in firmware
- Extract & manage resulting entropy using known approaches

Additional research needed: performance under jamming?

Conclusion

Lessons learned:

1. Always check quality of RNG
2. Let AP ignore group-addressed frames

3. Don’t put “expository” security algo’s in a specification
4. Don’t transmit sensitive data before downgrade detection

Predicting and Abusing WPA2/802.11 Group Keys

Mathy Vanhoef - @vanhoefm

Questions?