Defeating MAC Address Randomization Through Timing Attacks

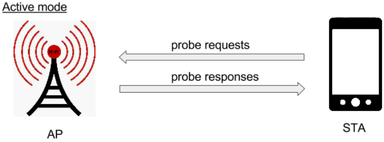
Célestin Matte[†], Mathieu Cunche[†], Franck Rousseau[‡], Mathy Vanhoef[II]

[†] Univ Lyon, INSA Lyon, Inria, CITI, France, Région Rhône-Alpes funding, [‡] Grenoble Institute of Technology, LIG, France, [II] iMinds-Distrinet, KU Leuven

WiSec'16 - July 18th 2016

Introduction - Wi-Fi service discovery

- Wi-Fi stations discover APs by sending `probe request` frames.
  - Containing a unique identifier: the MAC address



Introduction - Tracking

- What: getting the knowledge of a device's presence over time
- Who: businesses, intelligence services, nasty neighbours, employers...
  - Many retail tracking start-ups: Nomi, Euclid, Purple WiFi...
- Privacy issue: no consent nor awareness

Introduction

- MAC address randomization proposed to prevent tracking
  - Being deployed in major OSes
    - iOS 8, Android 6, Windows 10, Linux kernel 3.18
- Is it enough to prevent tracking ?
- No: Fingerprints can be built using the content of probe requests[1]
- Is it even necessary?
- We show that:
  - Randomization can be defeated using an attack based on the timing of probe requests
  - Fingerprints built using this attack are consistent over time

[1] Mathy Vanhoef et al. "Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms". In: *AsiaCCS*. May 2016. URL: https://hal.inria.fr/hal-01282900.

MAC address randomization scheme

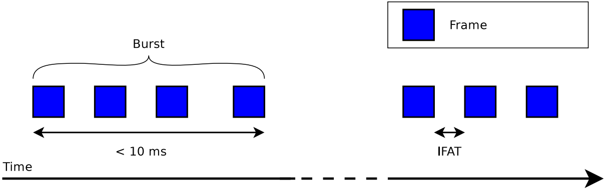
- MAC address randomization: many different implementations
- How often does the MAC address change?
  - Linux: Every few bursts
  - iOS > 9: every few bursts (2-4), sometimes for every burst
- $\rightarrow$ Can we build a fingerprint with so few frames?

Introduction - attacker model

- Attacker capabilities
  - Monitoring wireless channels
  - Single channel, single location
- Attacker objectives
  - Tracking devices
  - $\equiv$ Group frames belonging to the same device

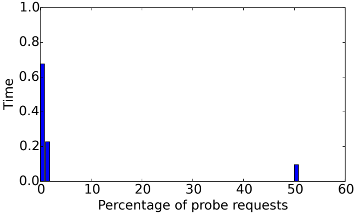
Terminology

- Burst: group of probe requests sent within 10ms
- Burst set: group of bursts sent with the same MAC address
- IFAT: Inter-Frame Arrival Time



Signature[2]

- Cut time into bins
- For each bin, store percentage of IFATs + average IFAT value



[2] Jason Franklin et al. "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting.". In: *Usenix Security*. Vol. 6. 2006.

Distances

- We need to define distances between temporal distributions
- $D1$: basic distance[3]

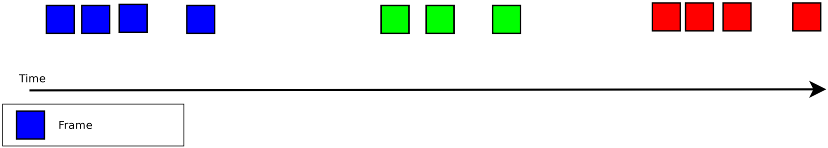
$$D1_{AB} = \sum_{b \in \mathcal{B}} (|P_b^B - P_b^A| + \frac{(P_b^A + P_b^B)}{2} * |M_b^B - M_b^A|)$$

- $D2$: add inter-burst set arrival time
- $D3$: hybrid distance between $D1$ and $D2$ (do not give strong credit to inter-burst set arrival time)

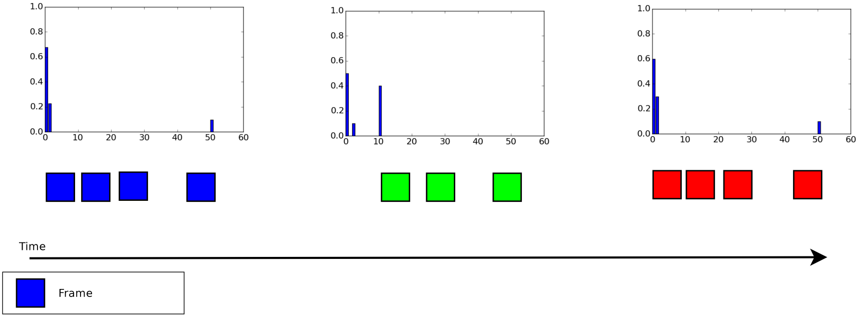
[3] Jason Franklin et al. "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting.". In: *Usenix Security*. Vol. 6. 2006.

Algorithm

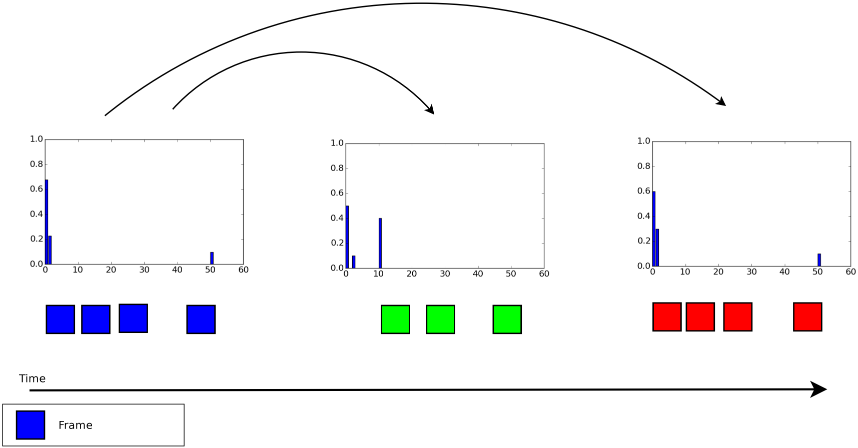
Algorithm



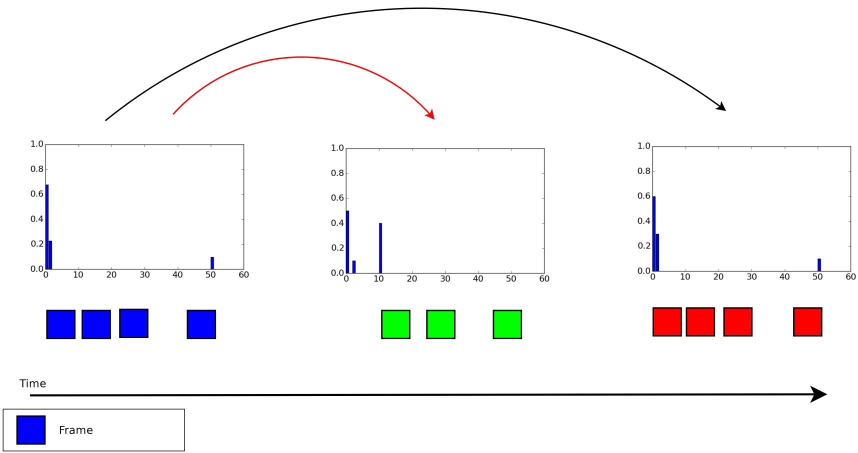
Algorithm



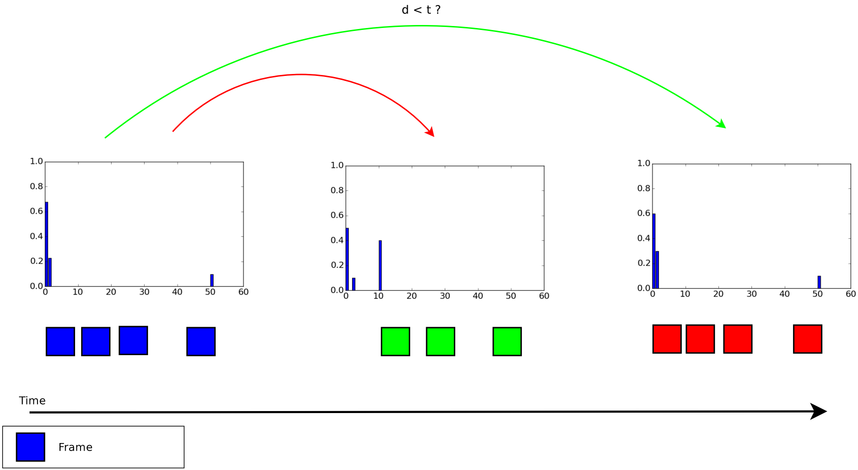
Algorithm



Algorithm



Algorithm



Experiments and results - dataset

- Dataset: 120 000 probe requests sent by 550 devices @lab, 6 days
- Simulate random MAC addresses

Experiments and results - metrics

- Accuracy: ratio of correct decisions
- TPR: number of burst sets from devices using random MAC addresses correctly grouped together, over the number of burst sets from devices using random MAC addresses
- FPR: number of burst sets incorrectly grouped with burst sets from other devices, over the total number of burst sets

Experiments and results - performance

- (after parameters selection)

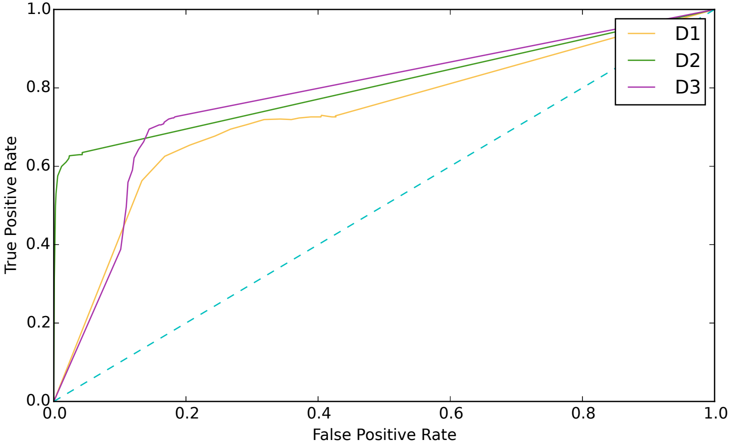


Figure: ROC curve of the three distances, over the range of threshold values.

Results summary

(After parameters selection)

Table: Results of the attack with the best parameters and options.

Distance	Accuracy	TPR	FPR
D1	66.8%	74.1%	24.3%
D2	**77.2%**	64.0%	**0.6%**
D3	71.8%	**75.2%**	17.5%

Countermeasures

- Changing the MAC address more often, every burst/frame
- Random delay between probe and between bursts

Conclusion

Context:

- MAC address randomization during Wi-Fi service discovery deployed to prevent tracking
- Is it enough?

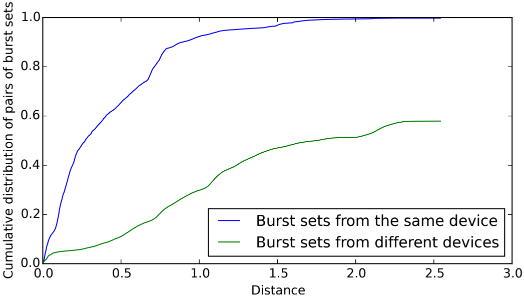
We showed that:

- Randomization can be defeated using an attack based on the timing of probe requests
- Fingerprints built using this attack are consistent over time

Discussion:

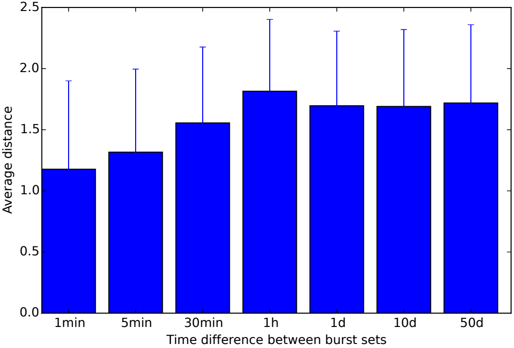
- The content of the probe requests is not even necessary to track devices

- Distance of probe requests from same device vs. distance of probe requests from different devices



Experiments and results - Stability

- Select probe requests separated by chosen time difference
- Compute distance



Algorithm

Algorithm 1: Random MAC breaking

Input: $\mathcal{G}$: groups of burst sets, grouped by MAC address
t: distance threshold
d: a distance function
Returns: $\mathcal{A}$: dictionary of aliases

$\mathcal{A} \leftarrow \emptyset$
$\mathcal{D} \leftarrow \emptyset$ `// Database of signatures`

Algorithm

Algorithm 2: Random MAC breaking

Input: $\mathcal{G}$: groups of burst sets, grouped by MAC address
t: distance threshold
d: a distance function
Returns: $\mathcal{A}$: dictionary of aliases

$\mathcal{A} \leftarrow \emptyset$
$\mathcal{D} \leftarrow \emptyset$ `// Database of signatures`
foreach $\mathcal{B} \in \mathcal{G}$ **do**

end

Algorithm

Algorithm 3: Random MAC breaking

Input: $\mathcal{G}$: groups of burst sets, grouped by MAC address
t: distance threshold
d: a distance function

Returns: $\mathcal{A}$: dictionary of aliases

$\mathcal{A} \leftarrow \emptyset$
$\mathcal{D} \leftarrow \emptyset$ `// Database of signatures`
foreach $\mathcal{B} \in \mathcal{G}$ **do**
  $\mathcal{S} \leftarrow$ signature($\mathcal{B}$)
end

Algorithm

Algorithm 4: Random MAC breaking

Input: $\mathcal{G}$: groups of burst sets, grouped by MAC address
t: distance threshold
d: a distance function
Returns: $\mathcal{A}$: dictionary of aliases

```
A ← ∅
D ← ∅                              // Database of signatures
foreach B ∈ G do
    S ← signature(B)
    d_min ← min(d(S, S')) where S' ∈ D
end
```

$\mathcal{A} \leftarrow \emptyset$
$\mathcal{D} \leftarrow \emptyset$ // Database of signatures
foreach $\mathcal{B} \in \mathcal{G}$ **do**
  $\mathcal{S} \leftarrow$ signature($\mathcal{B}$)
  $d_{min} \leftarrow min(d(\mathcal{S}, \mathcal{S}'))$ where $\mathcal{S}' \in \mathcal{D}$)
end

Algorithm

Algorithm 5: Random MAC breaking

Input: $\mathcal{G}$: groups of burst sets, grouped by MAC address
t: distance threshold
d: a distance function
Returns: $\mathcal{A}$: dictionary of aliases

```
A ← ∅
D ← ∅                                    // Database of signatures
foreach B ∈ G do
    S ← signature(B)
    d_min ← min(d(S, S′) where S′ ∈ D)
    if d_min < t then
        A[B.mac] ← A[S′.mac]                             // Alias
    else
        A[B.mac] ← B.mac                       // New MAC address
    end
end
```

$\mathcal{A} \leftarrow \emptyset$
$\mathcal{D} \leftarrow \emptyset$ // Database of signatures
foreach $\mathcal{B} \in \mathcal{G}$ **do**
  $\mathcal{S} \leftarrow$ signature($\mathcal{B}$)
  $d_{min} \leftarrow min(d(\mathcal{S}, \mathcal{S}'))$ where $\mathcal{S}' \in \mathcal{D}$)
  if $d_{min} < t$ **then**
    $\mathcal{A}[\mathcal{B}.mac] \leftarrow \mathcal{A}[\mathcal{S}'.mac]$ // Alias
  else
    $\mathcal{A}[\mathcal{B}.mac] \leftarrow \mathcal{B}.mac$ // New MAC address
  end
end

Algorithm

Algorithm 6: Random MAC breaking

Input: $\mathcal{G}$: groups of burst sets, grouped by MAC address
t: distance threshold
d: a distance function
Returns: $\mathcal{A}$: dictionary of aliases

```
A ← ∅
D ← ∅                                  // Database of signatures
foreach B ∈ G do
    S ← signature(B)
    d_min ← min(d(S, S′) where S′ ∈ D)
    if d_min < t then
        A[B.mac] ← A[S′.mac]                         // Alias
    else
        A[B.mac] ← B.mac                   // New MAC address
    end
    D ← D ∪ S
end
return A
```