



# Cracking WiFi at Scale with One Simple Trick

Ido Hoorvitch 10/26/21



*How I Cracked 70% of Tel Aviv's Wifi Networks (from a Sample of 5,000 Gathered Wifi).*

In the past seven years that I've lived in Tel Aviv, I've changed apartments four times. Every time I faced the same scenario: the internet company took several days to connect the apartment, leaving me disconnected and frustrated while trying to watch laggy Netflix on the TV with my cellphone hotspot. A solution I have to this scenario is having the "Hello. I am the new neighbor" talk with the neighbors while trying to get their cell phone number in case of emergencies — and asking if I could use their WiFi until the cable company connected me. I think we all can agree that not having internet easily falls into the emergency category! Often, their cell phone number was also their WiFi password!

I hypothesized that most people living in Israel (and globally) have unsafe WiFi passwords that can be easily cracked or even guessed by curious neighbors or malicious actors.

The combination of my past experience, a relatively new WiFi attack that I will explain momentarily, a new monster cracking rig (8 x QUADRO RTX 8000 48GB GPUs) in CyberArk Labs and the fact that WiFi is everywhere

because connectivity is more important than ever drove me to research, whether I was right with my hypothesis or maybe just lucky.

With the continued shift to remote work due to the pandemic, securing home networks has become imperative and poses a risk to the enterprise if not done so. Home networks rarely have the same controls as enterprise networks. And a security program is only as strong as its weakest link.



Figure 1- CyberArk Labs new cracking rig

To test this hypothesis, I gathered 5,000 WiFi network hashes as my study group by strolling the streets in Tel Aviv with WiFi sniffing equipment. At the end of the research, I was able to break more than **70%** of the sniffed WiFi networks passwords with relative ease. The Tel Aviv Metropolitan area has more than 3.9 million people — you can imagine what the numbers would have been had we not cut our research off at 5,000 WiFi networks. And while this research was conducted in Tel Aviv, the routers that were susceptible to this attack — from many of the world’s largest vendors — are used by households and businesses worldwide.

In this blog, I demonstrate how easily (you do not need a cracking rig) and with little equipment unsecure WiFi passwords can be cracked, thus hacking the WiFi network .At the end, we will reveal statistics of the cracked hashes and explain how to defend your network from this type of

attack. Therefore, it is of utmost importance that we know and understand the cracking method to form an adequate defense.

## Let's dig in

Before [Jens "atom" Steube's](#) (*Hashcat's* lead developer) [research](#), when a hacker wanted to crack a WiFi password, they needed to capture a live four-way handshake between a client and a router occurring only during the establishment of the connection. Simply put, the attacker would need to be monitoring the network at the time the user or device connects to the WiFi network. Therefore, a hacker needed to be in a physical location between the access point (router) and the client, hoping that the user would enter the right password and that all four packets of the handshake were sniffed correctly. If a hacker did not want to wait until a victim establishes a connection (which can take hours, who connects to their home network while they are at work?), the attacker could de-authenticate an already-connected user to force the victim to have a new four-way handshake.

Another attack vector is to set up a malicious twin network with the same SSID (network name), hoping that the victim would try to log in to the fake network. A major shortcoming of this is, of course, that it is very noisy (meaning it can be easily traced) and can be easily noticed.

In simple English, if an adversary wanted to hack/crack a WiFi password, they need to be in the right place (between users and a router) at the right time (when users log in) and be lucky (users entered the correct password and all four packets were sniffed correctly).

All of this changed [with atom's groundbreaking research](#), which exposed a new vulnerability targeting *RSN IE* (Robust Security Network Information Element) to retrieve a *PMKID* hash (will be explained in a bit) that can be used to crack the target network password. *PMKID* is a hash that is used for roaming capabilities between APs. The legitimate use of *PMKID* is, however, of little relevance for the scope of this blog. Frankly, it makes little sense to enable it on routers for personal/private use (WPA2-personal), as usually there is no need for roaming in a personal network.

Atom's technique is clientless, making the need to capture a user's login in real time and the need for users to connect to the network at all **obsolete**. Furthermore, it only requires the attacker to capture a single frame and eliminate wrong passwords and malformed frames that are disturbing the

cracking process.

Plainly put, we do not need to wait for people connecting to their routers for this attack to be successful. We are just in the vicinity of the router/network getting a *PMKID* hash and trying to crack it.

To crack a *PMKID*, we first need to understand how it is generated.

### How is *PMKID* hash generated and what elements does it contain

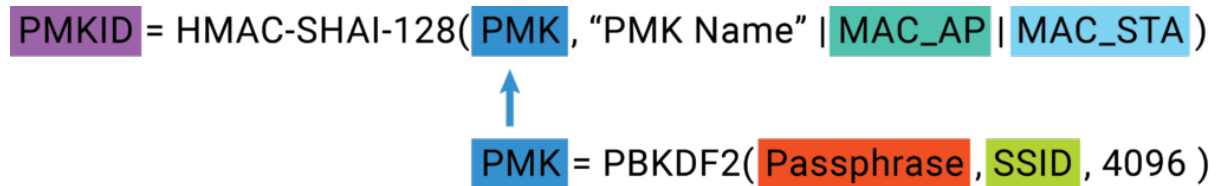
$$\text{PMKID} = \text{HMAC-SHA1-128}(\text{PMK}, \text{"PMK Name"} \mid \text{MAC\_AP} \mid \text{MAC\_STA})$$
$$\text{PMK} = \text{PBKDF2}(\text{Passphrase}, \text{SSID}, 4096)$$


Figure 2- Flow of Calculating PMKID hash and PMK

The hash calculation might seem daunting at first glance but let us dive into it.

We need to generate a *PMK* driven from *SSID* (the network name) and the *Passphrase*; then we generate a *PMKID* driven from the *PMK* we generated, the *AP MAC* address, and the *client MAC* address. So let us see where we can find those:

The PMK is computed as follows:

$$\text{PMK} = \text{PBKDF2}(\text{Passphrase}, \text{SSID}, 4096)$$

Figure 3- PMK calculation

- *Passphrase* – The WiFi password — hence, the part that we are really looking for.
- *SSID* – The name of the network. It is freely available at the router beacons (Figure 3).
- 4096 – Number of PBKDF2 iterations



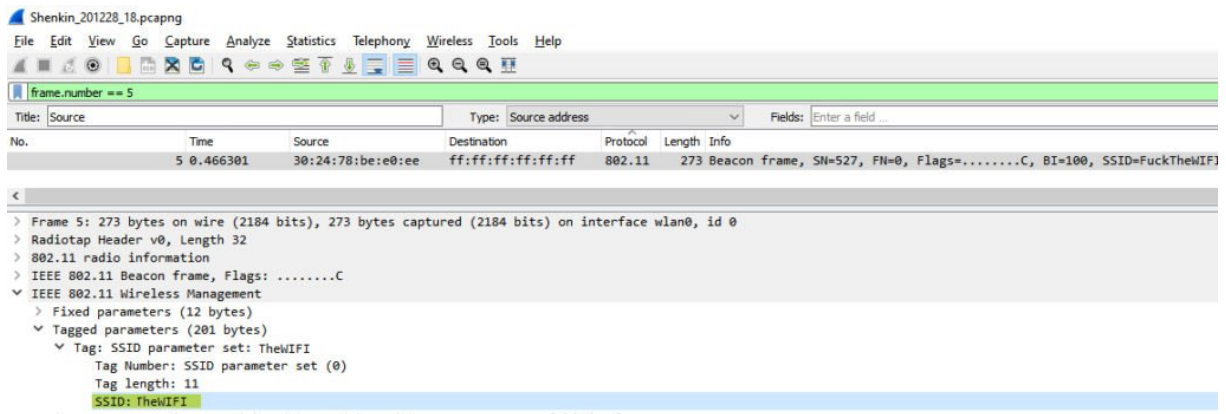


Figure 4 – SSID from a beacon

After a *PMK* was generated, we can generate a *PMKID*.  
The *PMKID* is computed as follows:

$$\text{PMKID} = \text{HMAC-SHA1-128}(\text{PMK}, \text{"PMK Name"} \mid \text{MAC\_AP} \mid \text{MAC\_STA})$$

Figure 5 – PMKID calculation

- **PMK** – What we are searching for, generated above. In WPA2 personal, the *PMK* is the *PSK* (*will be explained in the next paragraph*).
- *"PMK Name"* – Static string for all PKMIDs.
- **MAC\_AP** – Access Point's MAC address – This address can be found in any frame send by the router (Figure 4).
- **MAC\_STA** – The client's Mac address can be found in any frame sent by the client's computer(Figure 4). It can moreover be found in the output of *ifconfig\ip a* commands.

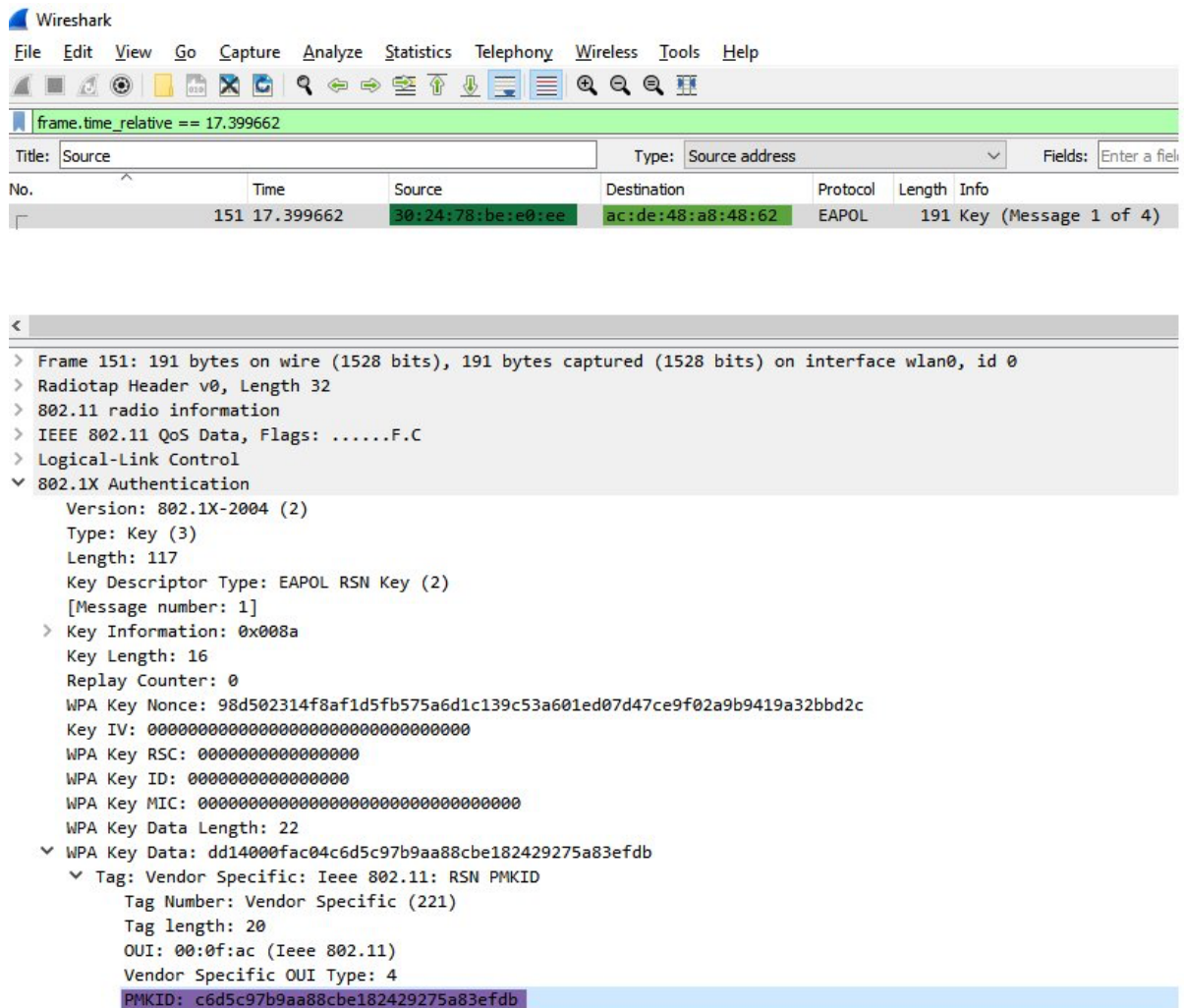


Figure 6 – PMKID, AP’s MAC, Client’s MAC

Cracking the **PMKID** hash is ultimately just generating/calculating *PMKs* with the **SSID** and different **passphrases**, then calculating **PMKID** from the **PMK** and the other information we obtained. Once we generated a **PMKID** equal to the **PMKID** that was retrieved from the AP (Figure 3), the hash is cracked; the **passphrases** that were used to generate the right **PMK** that the **PMKID** was generated from is the correct WiFi password.

Now we know how a *PMKID* is being generated, and we can continue to the sniffing and cracking phases of our research.

## Sniffing PMKID

To gather WiFi *PMKID* hashes, a wireless network interface that has monitor mode capabilities is required. Monitor mode allows packet capturing without having to *associate* with an access point.

I bought an AWUS036ACH [ALFA Network](#) card for \$50 (there are even cheaper options) that supports monitor mode and packet injection and went around the center of Tel Aviv to sniff WiFi.

**Before we can start the sniffing, we need to prepare our environment:**

I used an ubuntu machine with AWUS036ACH ALFA.



Figure 7 – AWUS036ACH ALFA NIC

We build the package [Hcxdumpool](#) — a great utility by *ZerBea* to capture packets from WLAN devices.

```
git clone https://github.com/ZerBea/hcxdumpool.git
sudo apt-get install libcurl4-OpenSSL-dev libssl-dev pkg-config
make
```

After that, we need to install drivers with monitor mode capability. Each chipset has its drivers:

```
git clone -b v5.6.4.2 https://github.com/aircrack-ng/rtl8812au.git
make && make install
```

It is recommended to shut down services that might interfere with *Hcxdumpool* execution:

```
sudo systemctl stop wpa_supplicant
```

```
sudo service NetworkManager stop
```

Then it is time to start sniffing. *Hcxdumpool* is a powerful tool that can be used for various attacks, not only the *PMKID*; therefore, we disable any attack that is not targeting *PMKID*.

```
sudo ../../tools/hcxdumpool/hcxdumpool -i wlx00c0caac2745 -o Wi-Fi_PMKID.pcapng --  
disable_deauthentication --disable_client_attacks --enable_status=3
```

- `-i` my *Alfa NIC*, you can execute `ifconfig\ip a` to find your interface name.
- `-o` the output *pcapng of the execution*.

*Now wear a hoody, because you will get a *PMKID* of every network you cross by that is vulnerable to the attack.*



Figure 8 – Me in a hoodie

When I reached 5,000 collected networks, I decided to quit; Israeli summer was too hot for me, so I turned to the fun part — cracking.

## It's cracking time!



Our first step in the cracking procedure is to install *hashcat*, the world's fastest and most advanced password recovery tool. As the sniffing results are in the form of *pcapng*, we needed to convert them into a hashfile format to fit it to *hashcat*. For this goal, I made use of another tool from the great suite of *hcxtools*.

```
hcxpcapngtool -o Wi-Fi_pmkid_hash_22000_file.txt Wi-Fi_PMKID.pcapng
```

Resulting in a *hashfile* that each line takes on the following structure:

```
SIGNATURE*TYPE*PMKID/MIC*MACAP*MACSTA*ESSID* * *
```

Here is an example of a hashline:

```
WPA*01*c6d5c97b9aa88cbe182429275a83efdb*302478bee0ee*acde48a8  
4862*54686557494649***
```

- SIGNATURE = "WPA"
- TYPE = 01 for PMKID, 02 for EAPOL, others to follow
- PMKID/MIC = PMKID if TYPE==01, MIC if TYPE==02
- MACAP = MAC of AP
- MACSTA = MAC of station
- ESSID = ESSID
- Not used in a PMKID attack:
  - ANONCE = ANONCE
  - EAPOL = EAPOL (SNONCE is in here)
  - MESSAGEPAIR = Bitmask

The next step is to commence the cracking procedure by executing *hashcat*:

Hashcat's capabilities include several cracking methods, of which the most common are dictionary + rules and mask attack. The methods differ in the way they are forming the *Passphrase*.

We chose to start with what's called a "mask attack," due to the terrible habit many people living in Israel have of using their cellphone numbers as WiFi passwords. You can think of mask attack as Regex:

- ?d – digits
- ?l – lower case characters
- ?u – Upper case characters
- ?s – special symbols as ? ! \$ .....

The mask for the password: 202!\$ummeR would become `?d?d?d?s?s?l?l?l?l?u`

Here is my Hashcat command that tried all the possible cellphone numbers combinations in Israel [the Israeli cellphone prefix is **05**]

```
sudo hashcat -a 3 -w4 -m 22000 /home/tuser/ashes/Wi-Fi_pmkid_hash_22000_file.txt  
05?d?d?d?d?d?d?d?d -o /home/tuser/ashes/pmkid_cracked.txt
```

During this first execution of the mask attack, we cracked **2,200 passwords**. Let's calculate the number of options for Israeli cellphone numbers:

It is 10 digits long and it starts with 05. Therefore, we need to guess the remaining 8 digits.

Each digit has 10 options (0-9), hence  $10^{**}8$  possible combinations. One hundred million seems like a lot of combinations, but our monster rig calculates at the speed of 6819.8 kH/s which translates into 6,819,000 hashes per second.

A cracking rig is not required as my laptop can get to 194.4 kH/s, which translates into 194,000 hashes per second. That equals more than enough computing power to cycle through the possibilities necessary to crack the passwords. Consequently, it took my laptop roughly 9 minutes to break a single WiFi password with the characteristics of a cellphone number.  $(10^{**}8)/194,000 = \sim 516$  (seconds)/60 =  $\sim 9$  minutes.

The cracking speed for hashtypes differs because of different hash functions and the number of iterations. For example, PMKID is very slow compared to MD5 or NTLM. Nonetheless, it is feasible to crack a PMKID hash if the attacker focuses on a specific network, and the password is not complicated enough.

Afterward, we executed a standard dictionary attack with the most common dictionary, [Rockyou.txt](#), and cracked more than 900 hashes. Here is a small glimpse into [Rockyou.txt](#) content:

```
123456 12345 123456789 password iloveyou princess 1234567 rockyou  
12345678 abc123 nicole daniel babygirl monkey lovely jessica 654321 michael  
ashley
```

**Let's look over the statistics of the cracked hashes:**

Cracked passwords by their length:

Password Length	Occurrences
10	2405
8	744
9	368
12	14
11	14
14	7
13	7
Sum	3,559

As you can see, except for the 10-digit password — which we had a tailored mask for — as the password length increased, the number of cracked passwords decreased. The lesson here? The longer the password, the better.

Top 4 masks for the cracked passwords:

Mask	Occurrences	Meaning
Mask	Occurrences	Meaning
?d?d?d?d?d?d?d?d?d?d	2349	10 digits
?d?d?d?d?d?d?d?d	596	8 digits
?d?d?d?d?d?d?d?d	368	9 digits

Mask	Occurrences	Meaning
?1?1?1?1?1?1?1?1	320	8 lower case letters
Sum	3,633	

We can see that cracked passwords most often fit a mask that contains only digits or only lower-case characters.

Not all routers support roaming features and are, therefore, not vulnerable to the PMKID attack. However, our research found that routers manufactured by many of the world's largest vendors are vulnerable.

As I estimated beforehand, the process of sniffing WiFis and the subsequent cracking procedures was a very accessible undertaking in terms of equipment, costs and execution.

The bottom line is that in a couple of hours and with approximately \$50, your neighbor or a malicious actor can compromise your privacy and much more if you don't have a strong password.

## Conclusion

In total, we cracked more than 3,500 WiFi network in and around Tel Aviv – 70% of our sample.

The threat of a compromised WiFi network presents serious risk to individuals, small business owners and enterprises alike. And as we've shown, when an attacker can crack more than 70% of WiFi networks in a major global city with relative ease, greater attention must be paid to protecting oneself.

At the most basic level, people who use your network take part of your bandwidth, which may slow down your internet experience. However, more consequential is that once attackers gain access to your network, they can launch various man-in-the-middle (MITM) attacks. That can lead to attackers gaining access to your important accounts, such as your bank account, your email account (which is everything in modern life) and compromising other sensitive credentials. This also further opens attack

vectors to your IoT devices like smart home equipment, smart TVs, security systems, etc.

For the small business, the risk lies in an attacker infiltrating a network and then moving laterally to high-value applications or data, such as a billing system or cashier.

Concerning the enterprise, it's possible for an attacker to gain initial access to a remote user's WiFi and then hop to the user's computer and wait for a VPN connection or for the user to go to the office and move laterally from there.

From a broader perspective, computers and other devices are usually not accessible from outside of the network because of NAT, but once an attacker is in the network, it facilitates a range of attack vectors.

### **How should I protect myself?**

1. **Choose a complex password.** A strong password should include at least one lower case character, one upper case character, one symbol, one digit. It should be at least 10 characters long. It should be easily remembered and hard to anticipate. Bad example: Summer\$021
2. Change the default username and password of your router.
3. Update your router firmware version.
4. Disable weak encryption protocols (as WEP or WPA1).
5. Disable WPS.

It's important to note that implementing multi-factor authentication (MFA) for personal WiFi is difficult and largely impractical for a personal WiFi and a non-technical consumer. It is also unlikely that MFA will be widely available for general consumer use cases in the near future.

I would like to give a big shout-out to [Atom](#) and [ZerBea](#) for their incredible work on this attack technique and for their work in general.

I hope you enjoyed this blog and that you will take the required steps to secure your WiFi network. And as a reminder, none of the passwords we cracked were used for unauthorized access to these WiFi networks or any other information accessible via these networks.