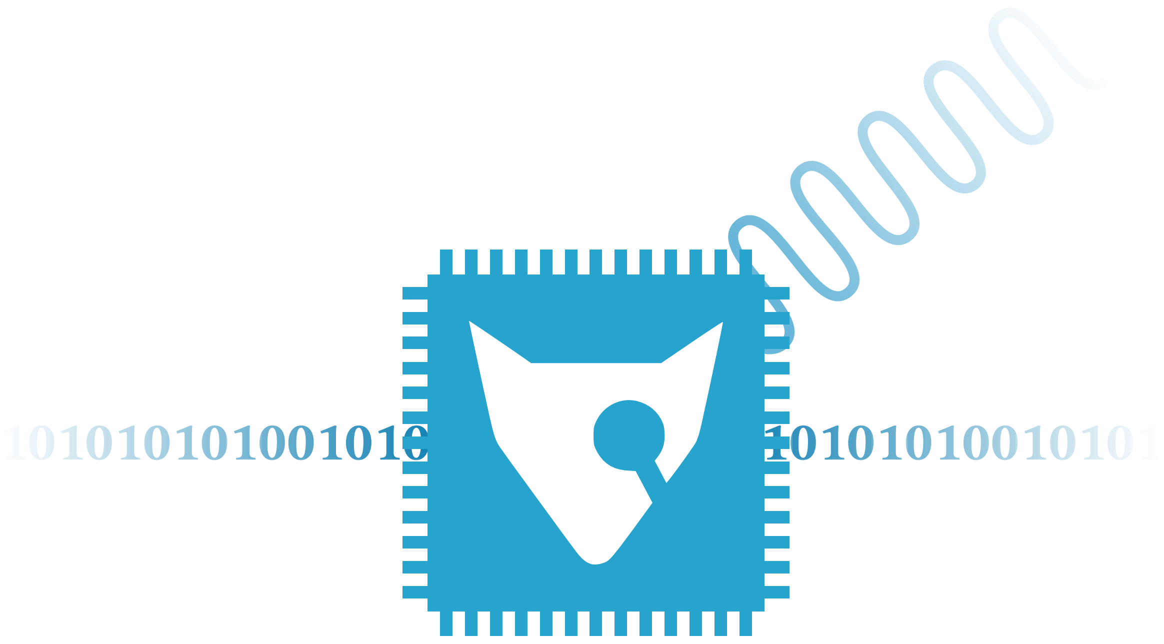


TEMPEST attacks against AES

Covertly stealing keys for €200

Overview

Side-channel attacks can recover secret keys from cryptographic algorithms (including the pervasive AES) using measurements such as power use. However, these previously-known attacks on AES tend to require unrestricted, physical access to the device. Using improved antenna and signal processing, Fox-IT and Riscure show how to covertly recover the encryption key from two realistic AES-256 implementations while:

1. Attacking at a distance of up to 1 *m* (30 *cm* in realistic conditions; "TEMPEST"),
2. Using minimal equipment (fits in a jacket pocket, costs less than €200) and
3. Needing only a few minutes (5 minutes for 1 *m* and 50 seconds for 30 *cm*.

To the best of our knowledge, this is the first public demonstration of such covert attacks from a distance. This demonstration reinforces the real need for defence-in-depth when designing high assurance systems — as Fox-IT is well known for.

Introduction

Some things are just inherent to electronic products. TEMPEST leakage is one of them. Remember:

> *Computers just work by pushing electrons around.*

This holds true for any appliance — whether programmed in JavaScript, C, or even a custom hardware design. Everything is eventually run on hardware (a.k.a. electronics). There are some interesting side-effects of this too. Paraphrasing our other Scottish friend, James Clerk Maxwell:

> *Pushing electrons around will induce a magnetic field*

So, running an algorithm will produce some fluctuating magnetic field. There is often nothing preventing us from measuring this EM radiation from close by, or even covertly from a distance. In fact, this is a technique which is relied upon in all wireless communication. All FM radio, WiFi, and GSM communications work by deliberately exploiting this! *TEMPEST attacks* measure this field from a distance and extract some useful information from it.

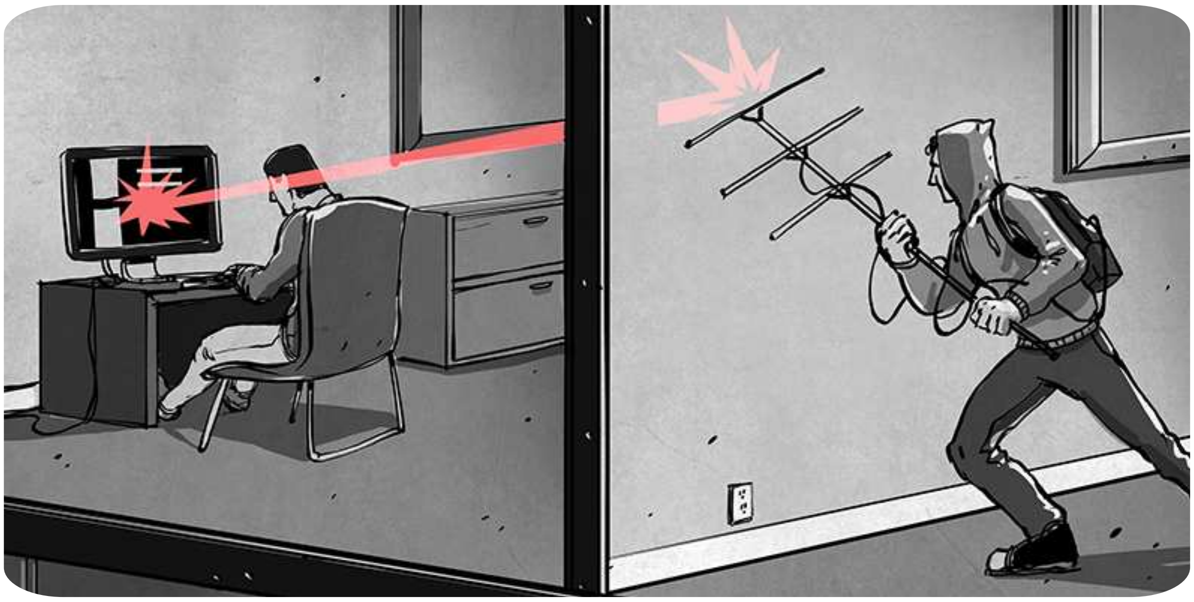


Image from [1]

Why are we interested?

Fox-IT's High Assurance Research & Development department has years of experience in building custom TEMPEST-certified appliances to protect state secrets.

This research turns the tables and focuses on attacking rather than defending. It goes beyond classical TEMPEST analysis, which is based on determining signal strength at a given distance, and investigates attack techniques under conditions close to real world environments. This knowledge can be used to offer products with improved countermeasures, above and beyond any aging standards.

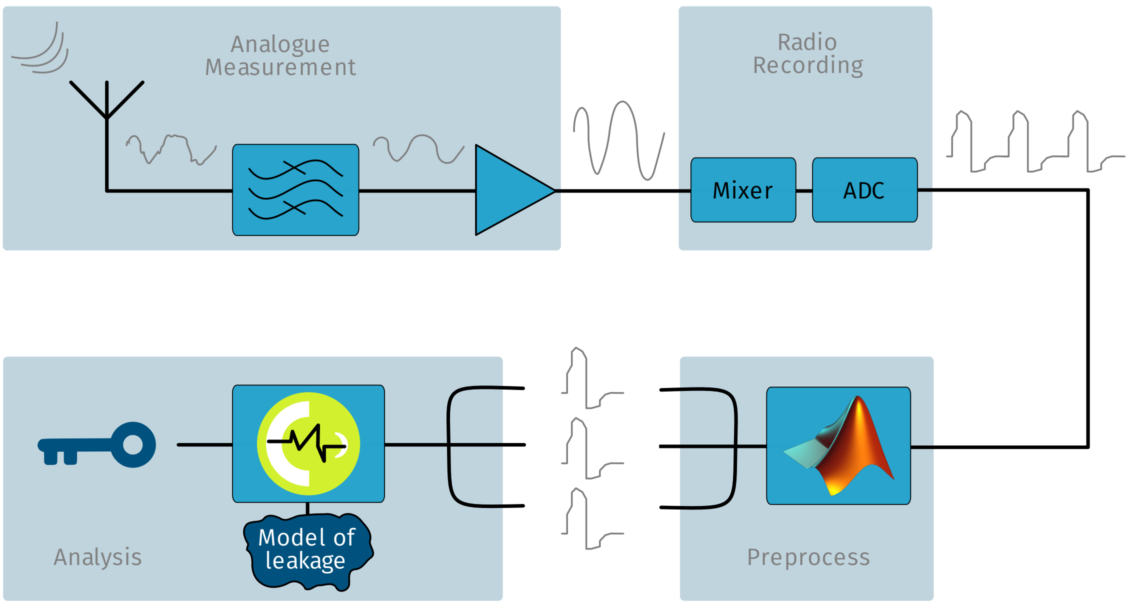
While there are published TEMPEST attacks against asymmetric crypto *[2][3]*, we have not seen any TEMPEST attacks against symmetric ciphers such as AES. These (excellent!) publications exploit the mathematical structure of asymmetrical algorithms to “amplify” the bit under attack into an easily-detected signal. However, AES does not have a structure that allows such amplification. We want to extend this work by considering sensible AES implementations. Our current attack requires that the attacker is able to observe (and ideally, choose) input or output data. This is entirely realistic for cryptographic devices with publicly visible output — e.g. network encryption appliances. So, let's have some fun...

Replicating near-field attack

A good first step is to develop our own setup to attack AES close by the chip. With support from Riscure, we got this up and running without many problems. The traditional setup records with an oscilloscope and an external signal for triggering. The attacker arms the oscilloscope, a signal from the board will trigger the start of the recording, and the measurement is copied over to the attacker's PC. This process is repeated for each block of encryption, until enough measurements have been collected for successful analysis.

We make use of radio hardware instead of this typical oscilloscope setup. This lets us record only the useful band of frequencies and ignore the rest. The data rate is now usually low enough to stream to directly to a PC which means we can capture much more information in a shorter time.

The figure below shows the workflow for the attack, grouped into 4 sections: analogue measurement, radio recording, preprocessing, and analysis.



Overview of the attack workflow

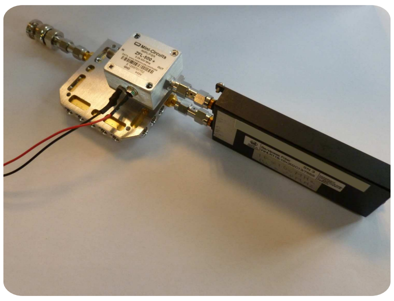
- Analogue measurement includes an antenna to measure the field, filtering around the attack frequency, and amplification.
- Radio recording can be any hardware capable of mixing our signal from a high clock frequency down to a more convenient one, and digitisation. More about the options here later.
- Preprocessing takes one large recording from the radio, detects where each encryption block starts, and splits the recording up into single encryption blocks. There is also some data conversion from complex (IQ) samples to a "real" signal for analysis.
- Analysis uses Riscure's Inspector tool to correlate between key byte guesses and the recorded traces to retrieve the secret key. The complex part here is to build a good model of how the device actually leaks information.

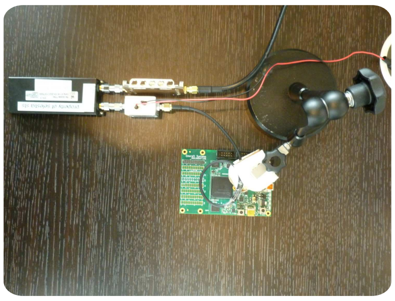
That was the setup in a vague, hand-waving sense, but what does it look like in practice?

Analogue measurement

The heart of the initial analogue front-end is small loop of wire. This small/magnetic loop antenna is amazingly simple to make — as we have done with a spare cable and some tape. The loop reacts to the fields from a nearby chip and we improve the signal with further filtering and amplification. We filter a 10 *MHz* band over the clock frequency (142 *MHz*) and amplify with cheap hardware readily available from Mini Circuits for under €200 total.

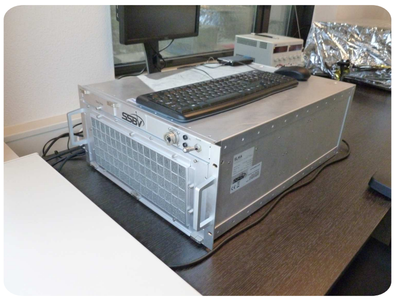
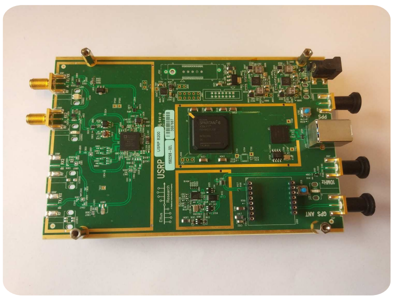
Loop antenna


External amplifier and bandpass filters

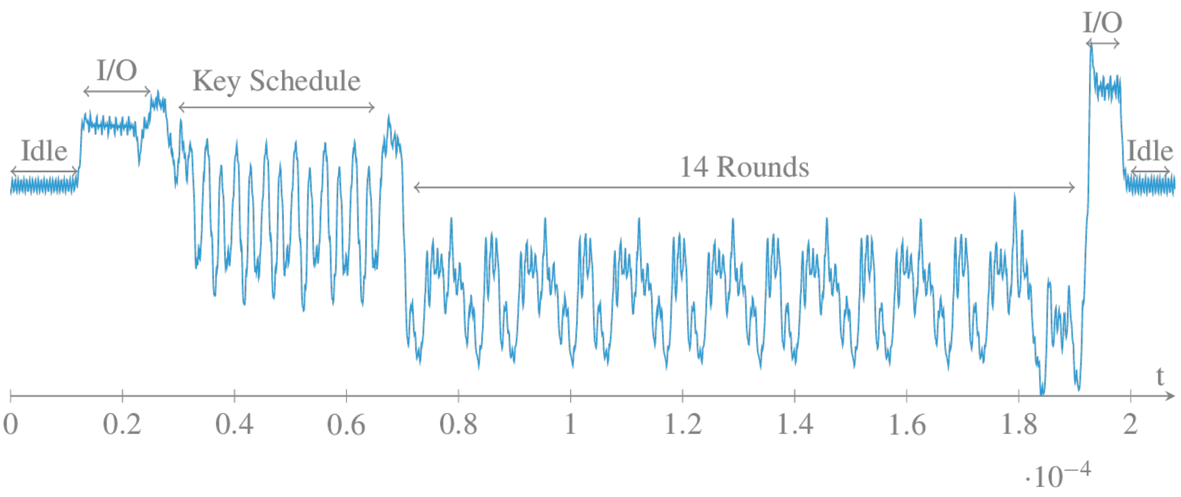

Example attack setup

Recording

The recording hardware can range from extremely high-end radio equipment, down to €20 USB SDRs. We have found that even the cheap USB dongles can be used to attack software implementations! This is not a game exclusively for nation states, but also anyone with pocket money and some free time. Our 3 go-to devices are compared below.

SR-7100 Data Recorder	USRP B200	RTLSDR
		
High-end	Low-end	Budget
€200k	€755	€20
500 MHz (max BW)	56 MHz	2.4 MHz
1.3 GB/s (max data rate)	184 MB/s	5.2 MB/s

For now, let's look at an averaged recording from the SR-7100 recorder using just 10 MHz bandwidth. The trace below shows our signal for one block of AES-256 encryption running on a SmartFusion2 target. We use OpenSSL's implementation of AES on the ARM Cortex-M3 core of the SmartFusion2. There are clear, distinct patterns for each stage of processing. We see I/O to and from the Cortex-M3, calculations for the key schedule, and the 14 encryption rounds.



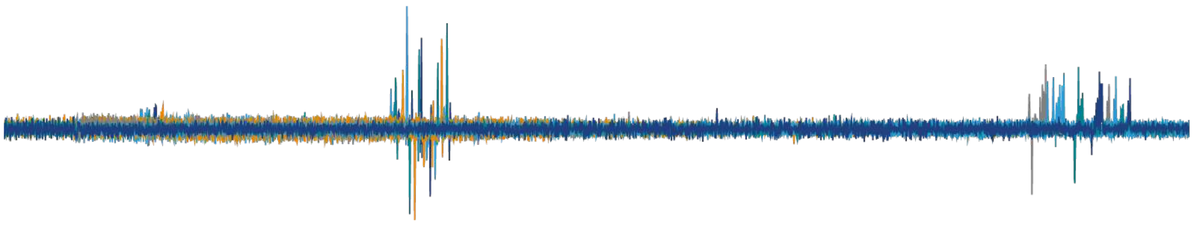
Overview trace showing pattern dependent on AES algorithm

So, we can measure a signal which is related to the instantaneous power consumption of part of the chip. This is still a long way from extracting secret keys though! To extract the key, we need to observe many different encryption blocks with different inputs and attempt to model how the device leaks information.

Analysis

> *Experts: we perform a standard attack with an S-box output (bitwise) model, and propose an address bus (Hamming distance) model if the firmware is available — feel free to skip to the next section.*

The recording we have captured is related to device power consumption. This is useful because part of the power consumption depends on the data the device is handling. It is this data dependence we exploit to extract the key. Let's quickly prove that we can "see" data using the recording. We take a set of different encryption blocks and correlate between either the (plaintext) input or (ciphertext) output data and our measurement traces. We do this by checking how well our measurements correlate with the number of "1" bits in the data (i.e. the data's Hamming weight). We would hope to get a plot with clear spikes where the input or output is processed. The plot below shows that we really *can* detect some data handled by the target.



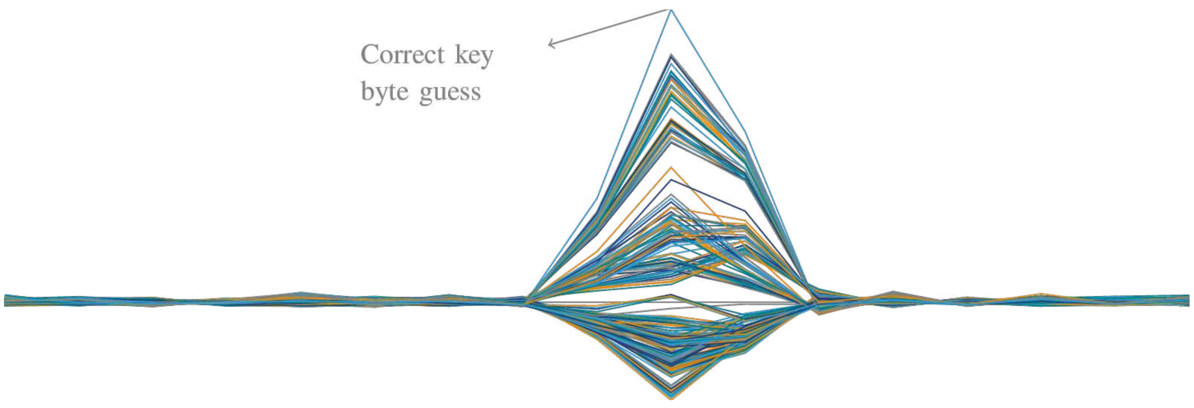
Correlation traces for I/O data (in 32-bit words)

Now we try to extract the secret key using a combination of guessing and correlation. Suppose we knew even 1 byte of the 32 byte key. We could start to predict some of the data handled inside the AES algorithm ("intermediates"). The correlation with our predicted intermediate data should be clear (similar to the I/O data graph above). Of course, as attackers, we do not know the key... but we can simply perform this prediction and correlation for all 256 possible values of 1 byte from the key. The value which provides the biggest peak in correlation should be the correct value. Using this approach only requires us to spend a few seconds guessing the correct value for each byte in turn (256 options per byte, for 32 bytes — so a total of 8192 guesses). In contrast, a direct brute-force attack on AES-256 would require 2^{256} guesses and would not complete before the end of the universe.

There are different intermediates we could use in our attack. The best intermediate to use will depend on the implementation, both in terms of software and hardware. OpenSSL uses lookups with "T tables", which is a sensible and common choice for 32-bit systems. The use of these T tables reduces the entire algorithm to a series of table lookups and exclusive ORs. Using the table lookups as our intermediates would provide some non-linearity. Non-linearity will produce a larger difference between the correct byte guess and a guess with 1 bit in error.

Especially if we can control the input data, we can successfully attack while using a very simple model. The results we present performs correlation for each bit of the data loaded from the T tables using a chosen input. We also go on to more accurately model the leakage. The improved model gives an appreciation for how the device leaks information but we avoid using it in practice due to the unreasonable level of target-specific knowledge required.

For the SmartFusion2 target, the leakage seems to largely originate from the AHB bus connecting the Cortex-M3 to the on-chip memories. We can correlate on the addresses of the T table lookups to attack the key. In this case, counting the number of "1" bits does not properly model the power use. Instead, we count the number of bits which transition between two sequential addresses on the bus (i.e. Hamming distance). This suggests that the bus logic briefly short-circuits when transitioning. As we are now working with the *transition* between two addresses, we must also know the previous/next address. With the I-cache disabled, this will likely be the address of a future instruction. With the I-cache enabled, this will likely be a previous explicit data load/store. Note that the SmartFusion2 does not have a D-cache in order to avoid cache coherency issues with the FPGA. The plot below shows the correlation traces for all 256 guesses of a key byte using our address line model.



Correlation traces for all key byte guesses (address Hamming distance model)

The correct byte does give the highest spike! This means we can recover the key (byte by byte) using only our measurements and the input data. Note that the other guesses fall into different bands. This is because our intermediate is linearly related to the key, so there are distinct groups of guesses which are each off by N bits.

Now we can extract the secret key from our target device, it is time to try it from a distance.

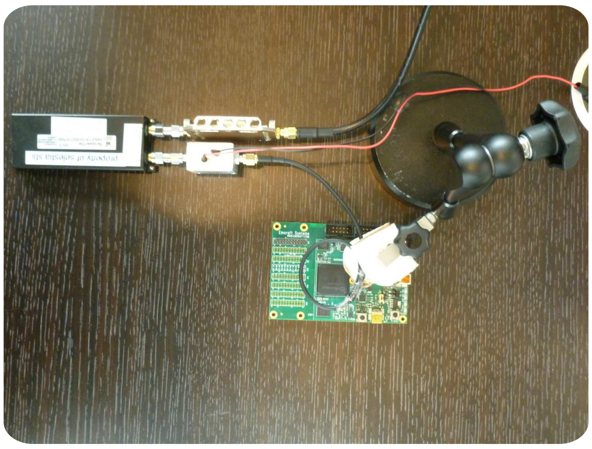
Gaining distance

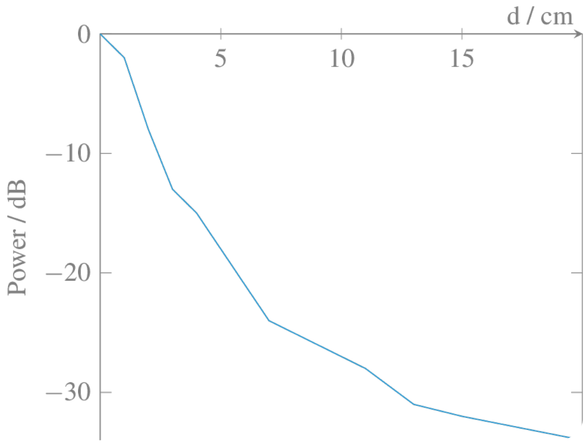
Performing the attack from a distance is where the real, new challenges lie. There is no reason why this should not work — but we believe this has not yet been shown in practice. We still measure the same physical effects, so the analysis remains the same. The changes to the setup are only with the analogue equipment.

Small loop

With the small loop antenna, the attack works from a few cm even with the €20 RTLSDR. This is a great solution for attacking through a sealed case without leaving any evidence of tampering (think smart energy meters and so). Depending on the target, sometimes the internal amplification of the RTL SDR is enough — then the attacker's full budget really can be €20!

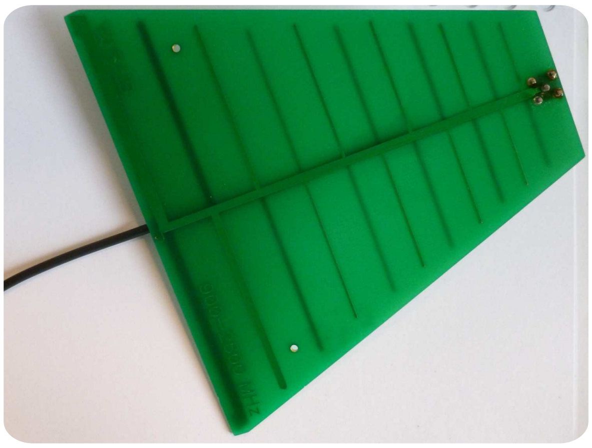
While these loops are a *very* easy way to reach a few cm, they are not likely to succeed from our goal of 1 *m*. We can see how the received power decays with distance below. The $1/x$ trend shows that our signal drops off very quickly, then gradually sinks into the noise floor.

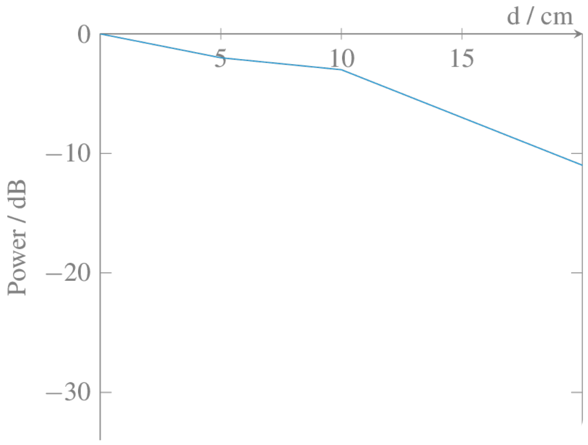




Log-periodic antenna

To improve the attack distance, we try a different style of antenna. The plot below shows the response for a PCB log-periodic antenna which is more directional. The received signal (in dB) drops off more linearly with distance, giving us a better chance of reaching our 1 *m* goal.





As these antennae tend to get larger as the intended frequency decreases, we initially test with a 400 *MHz* antenna to keep it small and cheap ($28). However, the SmartFusion2 can only be clocked up to 142 *MHz*. At this point, we switch to attack Xilinx's Pynq board. The software is still the same, we still attack the bus between ARM core and memory, but we can clock the memory at 400 *MHz*. We find switching to the new board surprising easy. Although the memory clock is ≈3 times faster (or effectively ≈6 as we now use DDR), we can still attack the Pynq board with the bandwidth supported by the RTL-SDR.

However, all of the analogue filtering was built for the 142 *MHz* signal. As a cheap and fast solution, we built a shielded chamber to reduce the effect of external signals using two boxes wrapped with emergency blankets.

Shielded box with emergency blankets

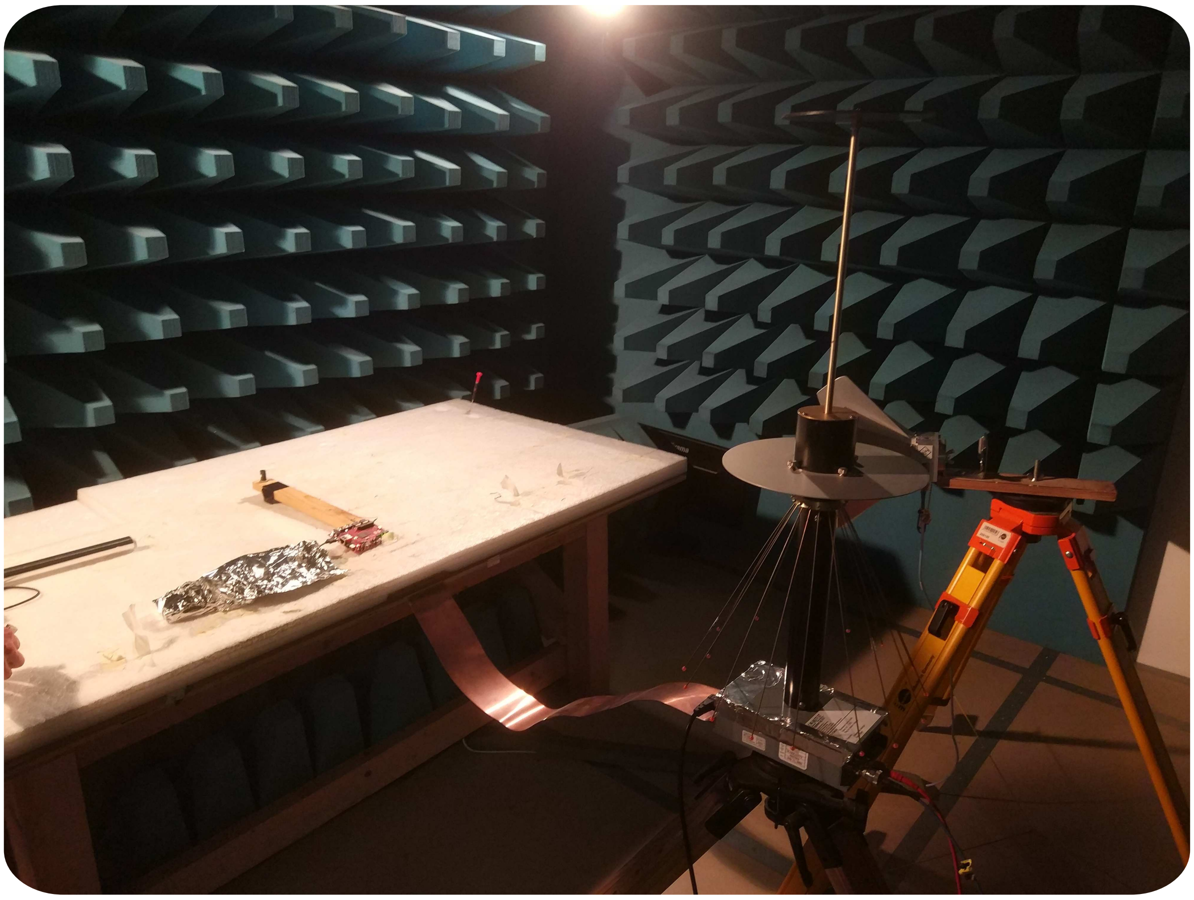
Attack setup inside box

This setup allowed us to successfully attack the device from 30 *cm*. This is using our budget recording setup (<€200) consisting of an RTL-SDR USB dongle, Mini Circuits amplifiers (ZFL-500+), and optional high-pass filters. We captured 400k traces for the attack in approximately 50 seconds. We have had limited time with this antenna so far, so the true maximum distance is yet to be determined. While still not at the 1 *m* goal, an attack from this distance with such a cheap setup is a nice step.

Ideal environment

We have found that it is possible to attack from distance in a DIY lab environment, but how far can we reach in an ideal environment?

We had the opportunity to use an anechoic chamber at OSPL, shown below. This provides an environment which is essentially free of external signals and most reflections of internal signals. We use a discone antenna (also seen below) to measure omnidirectionally with a very wide frequency range.



Setup at OSPL's anechoic chamber

In order to prove we are actually measuring from a distance, we must consider electrical isolation between the target device and the measurement setup:

- We power the target from a USB battery supply inside the chamber.
- The amplifiers are powered from a separate battery, outside the chamber.
- Our SmartFusion2 target needs to receive commands over Ethernet. In order to isolate this from our laptop and SDR, we convert from Ethernet to an optical signal before leaving the chamber. This also helps avoid introducing external signals back into the chamber over an Ethernet cable acting as an antenna.
- The Pynq target can operate without any Ethernet at all in a free-running loop with a known initialisation vector. The output is fed back the the input for each successive encryption.

Even with our <€200 (excluding antenna) recording setup we are able to successfully attack the Pynq board from 1 *m* distance. This is possible with 5 minutes of recording time which captures ≈2.4 million traces. This result shows that, under ideal conditions, attacks from 1 *m* are definitely possible.

Conclusion

Our work here has shown a proof of concept for TEMPEST attacks against symmetric crypto such as AES-256. To the best of our knowledge, this is the first public demonstration of such attacks. The low bandwidth requirements have allowed us to perform the attack with surprisingly cheap equipment (€20 radio, modest amplifiers and filters) at significant distances:

- 30 *cm* in a DIY lab environment with a 50 second recording
- 1 *m* in an ideal environment with a 5 minute recording

In practice this setup is well suited to attacking network encryption appliances. Many of these targets perform bulk encryption (possibly with attacker controlled data) and the ciphertext is often easily captured from elsewhere in the network. This again underscores the need for deep expertise and defence-in-depth when designing high assurance systems — contact our high assurance team if you need advice. Just following the TEMPEST norms can leave customers with an uncomfortably small safety margin.

By Craig Ramsay & Jasper Lohuis

> *We worked with many great people, and are thankful to all of them, but want to particularly thank:*
>
> - *Jos Haesakkers, Gerke Stoevelaar, Marco de Rooij, and Joachim Schipper from Fox-IT for their guidance*
> - *Rafael Boix Carpi, Bartek Gedrojc, Fred de Beer from Riscure and Chiel Spithout from OSPL for their expertise and generosity*
> - *Christo Butcher (Fox-IT), Louise Crockett and Robert Stewart (University of Strathclyde), and Christiaen Slot (Saxion University of Applied Science) for making this internship possible*

References

- [1] (*Image Source*) E. Williams,
"TEMPEST: a Tin Foil Hat for Your Electronics and Their Secrets",
October 2015
- [2] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer,
"Stealing Keys from PCs using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation",
Tel Aviv University, *February 2015*
- [3] D. Genkin, L. Pachmanov, I. Pipman, E. Tromer,
"ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs",
Tel Aviv University, *February 2016*