Side Channel Attacks on Smartphones and Embedded Devices using Standard Radio Equipment

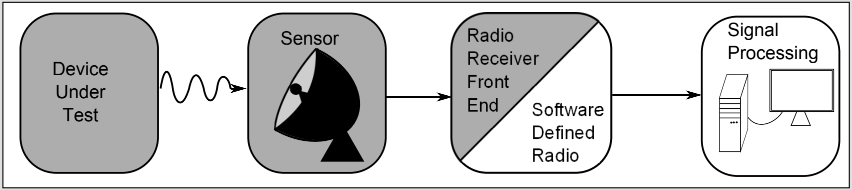
Gabriel Goller & Georg Sigl

14.4.2015

Giesecke & Devrient

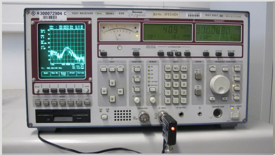
Creating Confidence.

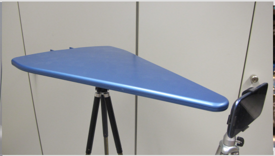
Introduction

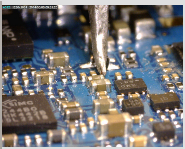


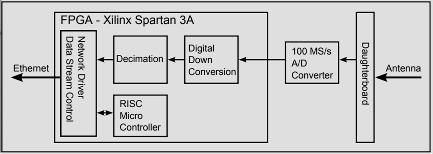
Capture the electromagnetic emanations of a device with state of the art radio equipment to use them for a side channel attack.

Experimental Setup - Hardware



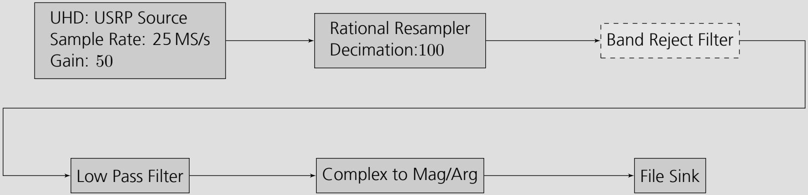






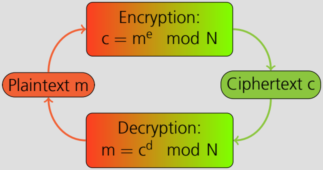
- 2 Antennas: Log-P and Bi-Quad
- ESN test receiver with preamplifier
- High-end setup using USRP N210 connected to IF of ESN
- DVB-T stick as low-cost alternative

Experimental Setup - Software



- GNURadio to process and record data
- Octave for offline post-processing

Device under Test - Software



Encryption: $c = m^e \mod N$

Ciphertext c

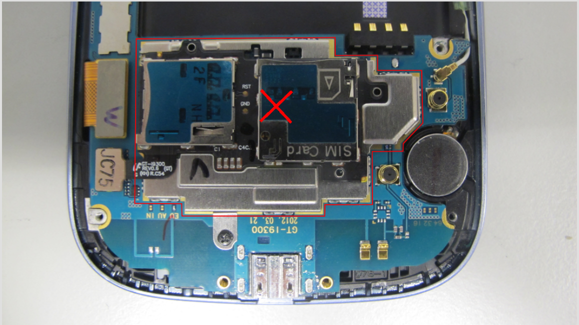
Decryption: $m = c^d \mod N$

Plaintext m

```
function square-and-multiply(c, d, N)
    result = 1
    for each bit(d)
            from (number_of_bits(d) - 1)
            downto 0
        result = square(result) mod N
        if bit(d) == 1
            result = (c * result) mod N
        end if
    end for
    return result
end function
```

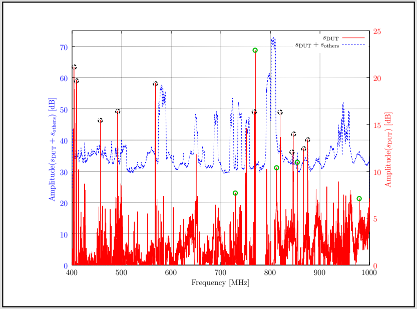
- Simple Square & Multiply Algorithm implemented with Android NDK using functions provided by OpenSSL.

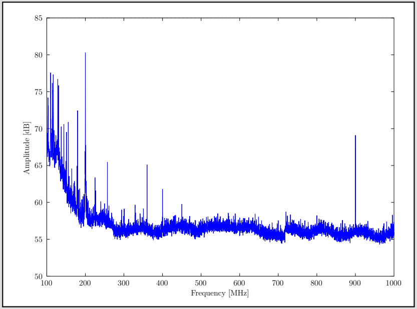
Device under Test - Hardware



- Android-based smartphone with ARM architecture
- Removed shielding plate for stronger emanations

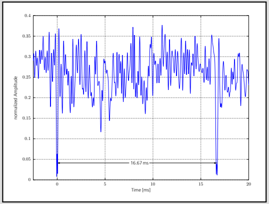
Finding Emanations

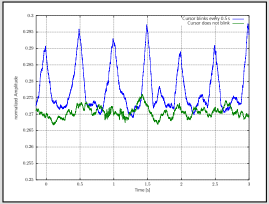


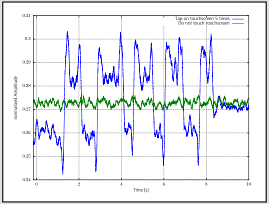


- Measurements using Frequency Sweep (left diagram)
- Measurements using Nearfield Probe (right diagram)
- Educated Guessing

Display Dependent

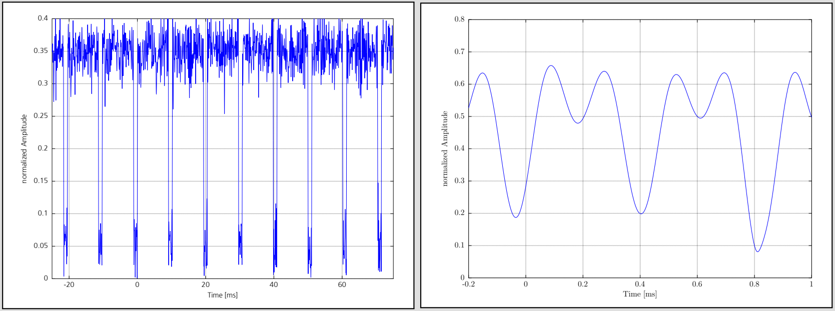






- Changes of display content and contact with display can be measured from a distance of $\sim 3\,\mathrm{m}$.
- No correlation with program flaw.

CPU Dependent

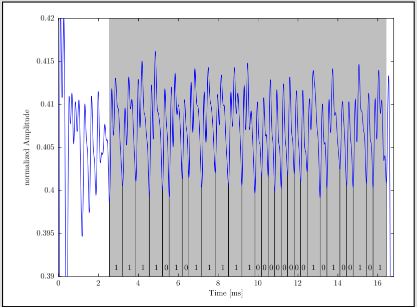


- A signal which correlates with the program flow can be found when the clock frequency of the CPU is set to a fixed value.
- No SPA possible.

Post-Processing of Signals

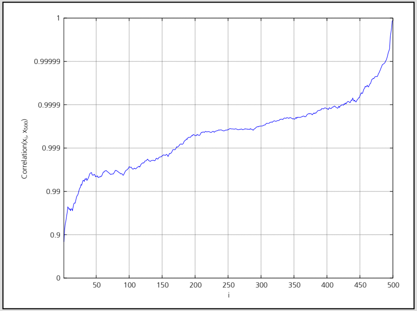
Steps:

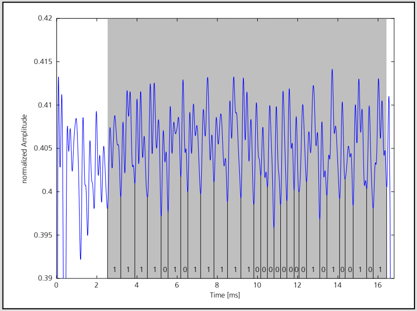
- Record signal with multiple S&M executions with same secret key d
- Extract each trace t where algorithm is executed (automated)
- Compute $y(t) = \mathrm{mean}(t_1(t), t_2(t), t_3(t), \ldots)$



Automated averaging of multiple signal blocks makes it possible to extract key of S&M algorithm.

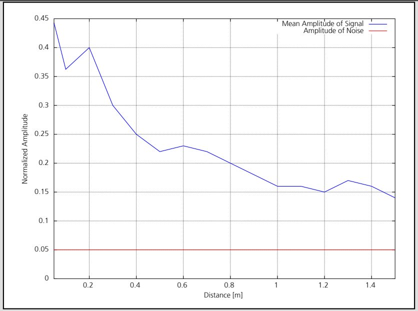
Evaluation - Number of Traces

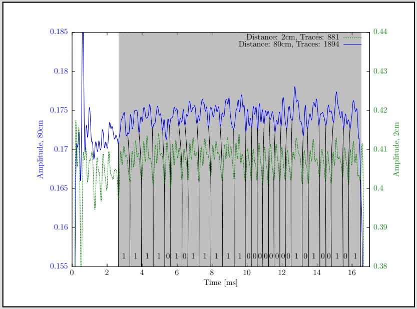




- $y(i) = \mathrm{corr}[\mathrm{mean}(t_1, t_2, \ldots, t_{500}), \mathrm{mean}(t_1, t_2, \ldots, t_i)]$
- ~ 170 traces are sufficient to reconstruct key

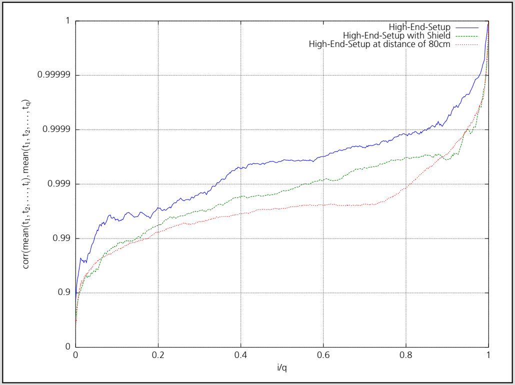
Evaluation - Distance & Shielding Plate





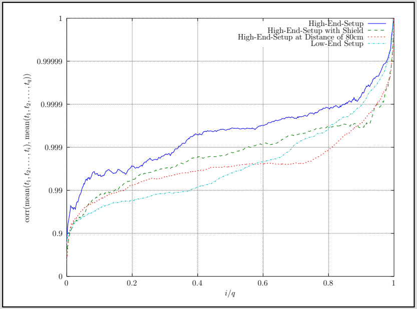
- Signal measurable up to a distance of 1.5 m.
- Number of traces increases, reconstruction succeeded at a maximal distance of 80 cm using 1894 traces.
- Reaffixing shielding plate results in similar effects.

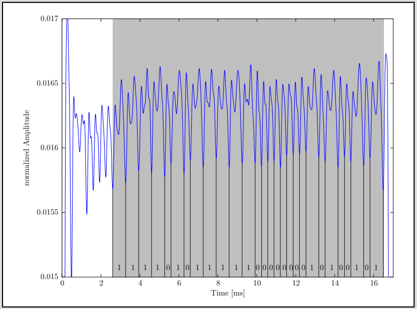
Number of Traces II



- Shielding: Correlation of 0.999 with 276 traces ($\approx$ factor 1.6)
- Distance: Correlation of 0.999 with 1530 traces ($\approx$ factor 9)

Evaluation - Lowcost Setup





- Reduced costs of under 30 €
- Signal-to-noise ratio decreased from 13.94 dB to 11.82 dB
- Correlation of 0.999 with 346 traces ($\approx$ factor 2)

Evaluation - Miscellaneous

Device	OS	CPU Frequency	Attack possible?	Remove Shielding?	Orientation
DUT 1 Smartphone	Android	900 MHz	Yes	Yes	→
DUT 2 Smartphone	Android	1000 MHz	Yes	No	↗
DUT 3 Smartphone	Android	1000 MHz	Yes	Yes	↑
DUT 4 SBC	Android	1000 MHz	Yes	No	→
DUT 5 SBC	Linux	900 MHz	Yes	No	→

- 5 different devices were tested, all with the same results.
- The smartphone also emits signals when disassembled.

Summary

- SCA on smartphones and embedded devices are feasible using standard radio equipment.
- The experimental setup can be built for less than 30 €.
- A private key can be extracted with only 170 traces.
- Attack was successfully conducted on multiple devices.

Demo - Lowcost Setup

```
function square–and–multiply(c, d, N)
    result = 1
    for each bit(d)
            from (number_of_bits(d) – 1)
            downto 0
        result = square(result) mod N
        if bit(d) == 1
        result = (c * result) mod N
        end if
        sleep()
    end for
    return result
end function
```