RSAConference2016
San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: CRYP-R02

ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs

Daniel Genkin
Technion,
Tel-Aviv University

Lev Pachmanov
Tel-Aviv University

Itamar Pipman
Tel-Aviv University

Eran Tromer
Tel-Aviv University

#RSAC

Key Extraction via Physical Side Channels

Small Devices

Big Devices

Modular Exponentiation (RSA, ElGamal)

Small Devices: [Fouque Kunz-Jacques Martinet Müller Valette 06] [Gandolfi Mourtel Oliver 01] [Homma Miyamoto Aoki Satoh Shamir 08] [Kocher 96] [Courrege Feix Rousselet 10] [Fouque Valette 03] [Kocher Jaffe Jum 99] [Messerges Dabbish Sloan 99] [Novak 02] [Walter Thompson 01] [Kühn 03]…

Big Devices:

- Acoustic [Genkin Shamir Tromer 14]
- EM, ground potential [Genkin Pipman Tromer 14]
- Cheap EM [Genkin Pachmanov Pipman Tromer 15]

Elliptic Curve Cryptography

Small Devices: [Cron 02], [Akishita Takagi 03], [Avanzi 05], [Biehl Meyer Müller 00], [Blömer Otto Seifert 06] [Ciet Joye 05] [Fouque Lercier Réal Valette 08] [Fouque Réal Valette Drissi 08] [Fouque Valette 03] [Goubin 02] [Herbst Medwed 09] [Itoh Izu Takenaka 08] [Karlof Wagner 03] [Medwed Oswald 09] [DeMolder Örs Preneel 07] [Okeya Sakurai 00] [Walter 04] …

Big Devices:

New Challenges

- Shorter keys, smaller numbers - even faster
- Different math

This Paper

Different scenario

- Not handed out to the adversary
- Attacker needs to be swift and inconspicuous

Speed

- 2GHz vs. 100MHz CPU
- Clock-rate attacks requires expansive and bulky equipment

Complexity & Noise

- Complex electronics running complicated software (in parallel)

Attacking ECDH:
GnuPG as a case study

Elliptic Curve Diffie-Hellman (ECDH) Encryption

- Standardized
  - OpenPGP [RFC 6637]
  - NIST SP800-56A
- Implementations
  - GnuPG (libgcrypt)
  - BouncyCastle
  - Google's end-to-end encrypted email

- Key Setup:
  - Secret key: random k
  - Public key: point $(k \cdot \mathbb{G})$

- Encryption:
  - Random number: k'
  - Ephemeral key: $t = KDF\left(k' \cdot (k \cdot \mathbb{G})\right)$
  - Ciphertext: $c = (AES_t(m), k' \cdot \mathbb{G})$

- Decryption:
  - Compute: $r = k \bullet (k' \cdot \mathbb{G})$
  - Obtain ephemeral key: $t = KDF(r)$
  - $m = AES_t(c')$

GnuPG's NAF representation

- Non-Adjacent Form (NAF) representation [Reitwiesner 60]
  - Allows positive and negative digits
  - $b = \Sigma_i 2^i b_i$ where $b_i \in \{-1,0,1\}$
  - Reduces the number of nonzero digits from ½ to ⅓
  - Example: $7=(0,\mathbf{1},\mathbf{1},\mathbf{1})_2=(\mathbf{1},0,0,\mathbf{-1})_2$

GnuPG's Scalar-by-Point Multiplication

```
point_mul(k, P) {
 A=P
 for i=n-1..0 do
D  A = 2*A
   if k[i]==1 then
A   A = A + P
   if k[i]==-1 then
I   P'= -P
A   A = A + P'
 return A
}
```

$A=[k_n\|\ldots\|k_{i+1}]\bullet P$

$A=[k_n\|\ldots\|k_{i+1}\| 0]\bullet P$

$A=[k_n\|\ldots\|k_{i+1}\| 1]\bullet P$

$A=[k_n\|\ldots\|k_{i+1}\|-1]\bullet P$

$A=[k_n\|\ldots\|k_{i+1}\| k_i]\bullet P$

GnuPG's Scalar-by-Point Multiplication

```
point_mul(k, P) {
  A=P
  for i=n-1..0 do
D   A = 2*A
    if k[i]==1 then
A     A = A + P
    if k[i]==-1 then
I     P'= -P
A     A = A + P'
  return A
}
```

measure → DADDI ADIA… → deduce → k=1,0,-1,-1,…

```
point_inverse(P) {
  P'.x = P.x
  P'.y = -P.y
  return P'
}
```

5MHz measurements
vs.
2000MHz CPU

GnuPG's Scalar-by-Point Multiplication

```
point_mul(k, P) {
 A=P
 for i=n-1..0 do
  A = 2*A
  if k[i]==1 then
   A = A + P
  if k[i]==-1 then
   P'= -P
   A = A + P'
 return A
}
```

Leakage self amplification

[GST14], [GPT14], [GPPT15]

abuse algorithm's own code to amplify its own leakage!

Craft suitable cipher-text to affect the inner-most loop

Small differences in repeated inner-most loops cause a big overall difference in code behavior

GnuPG's Scalar-by-Point Multiplication


```
point_mul(k, P) {
 A=P
 for i=n-1..0 do
  A = 2*A
  if k[i]==1 the
   A = A + P
  if k[i]==-1 then
   P'= -P
   A = A + P'
 return A
}
```

```
point_add(P1, P2){
 if P1.z==0 then
  return P2
 if P2.z==0 then
  return P1
 t1 = P1.x*(P2.z²)
 t2 = P2.x*(P1.z²)
 t3 = t1-t2
 t4 = P1.y*(P2.z³)
 t5 = P2.y*(P1.z³)
```

x00000001

x8e216f53a2...

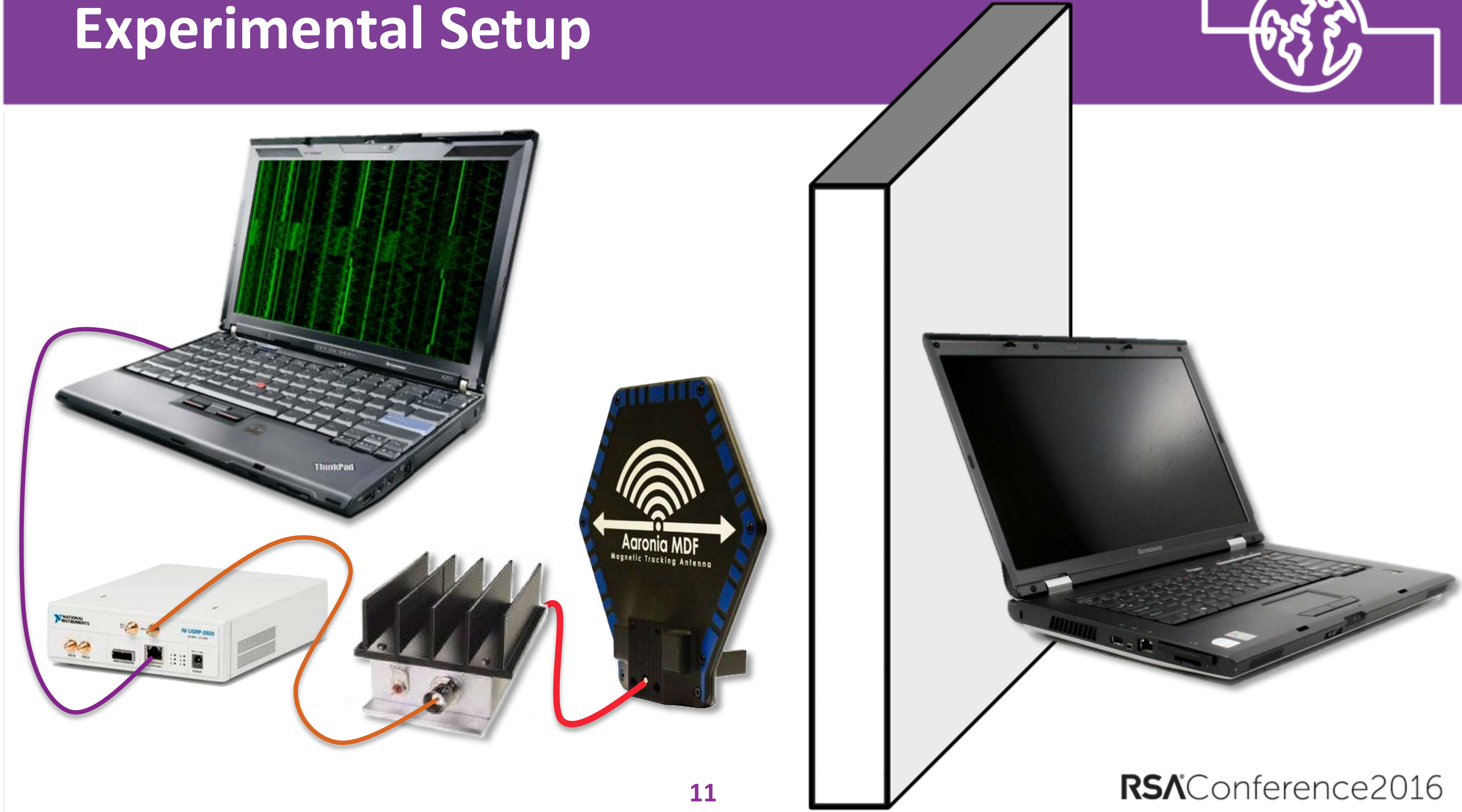
```
point_add(P1, P2){
 ...
 t5 = P2.y*(P1.z³)
 ...
}
```

if k[i]==1 then P2.y=1 so P2.y is short
if k[i]==-1 then P2.y=-1 so P2.y is long

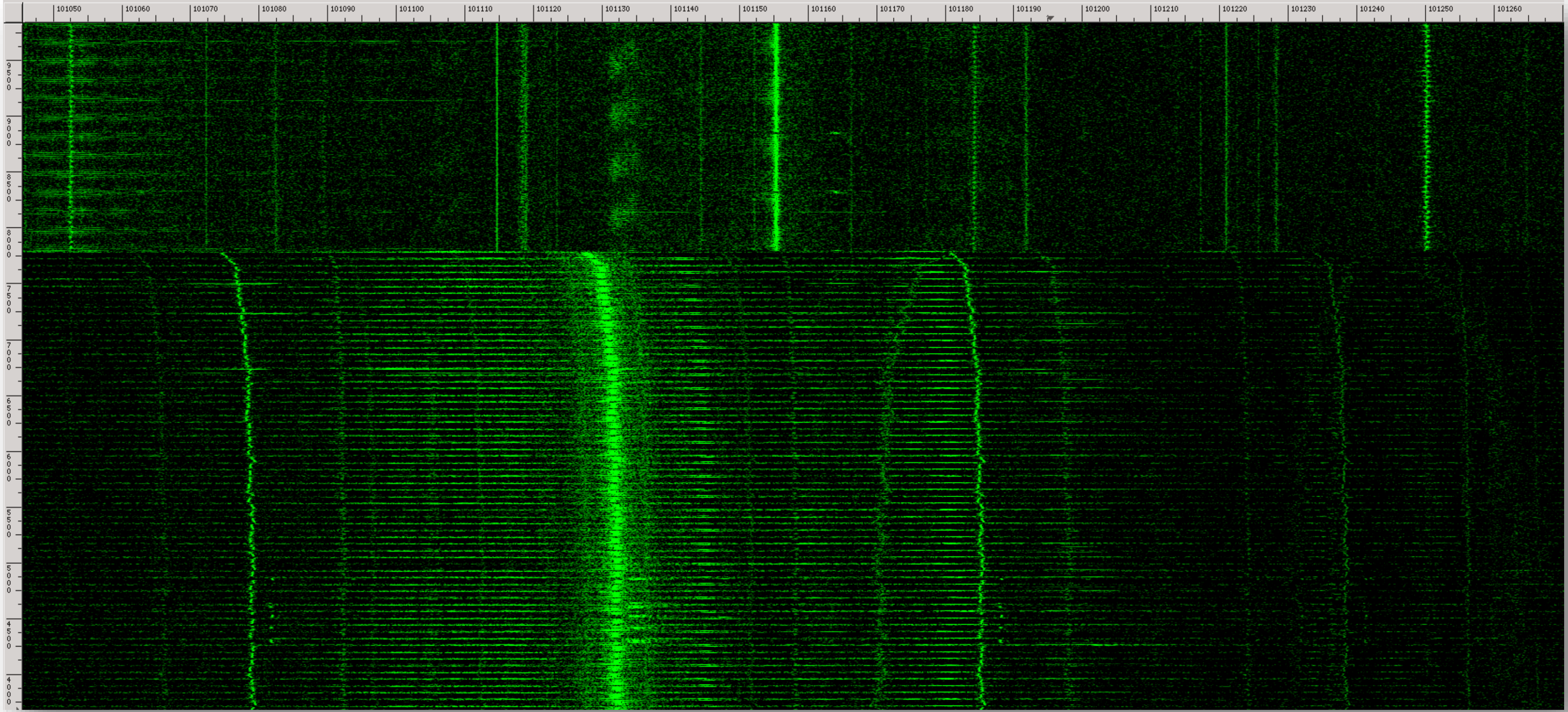
1041 µs
vs.
1110 µs

Live Demo

Experimental Setup

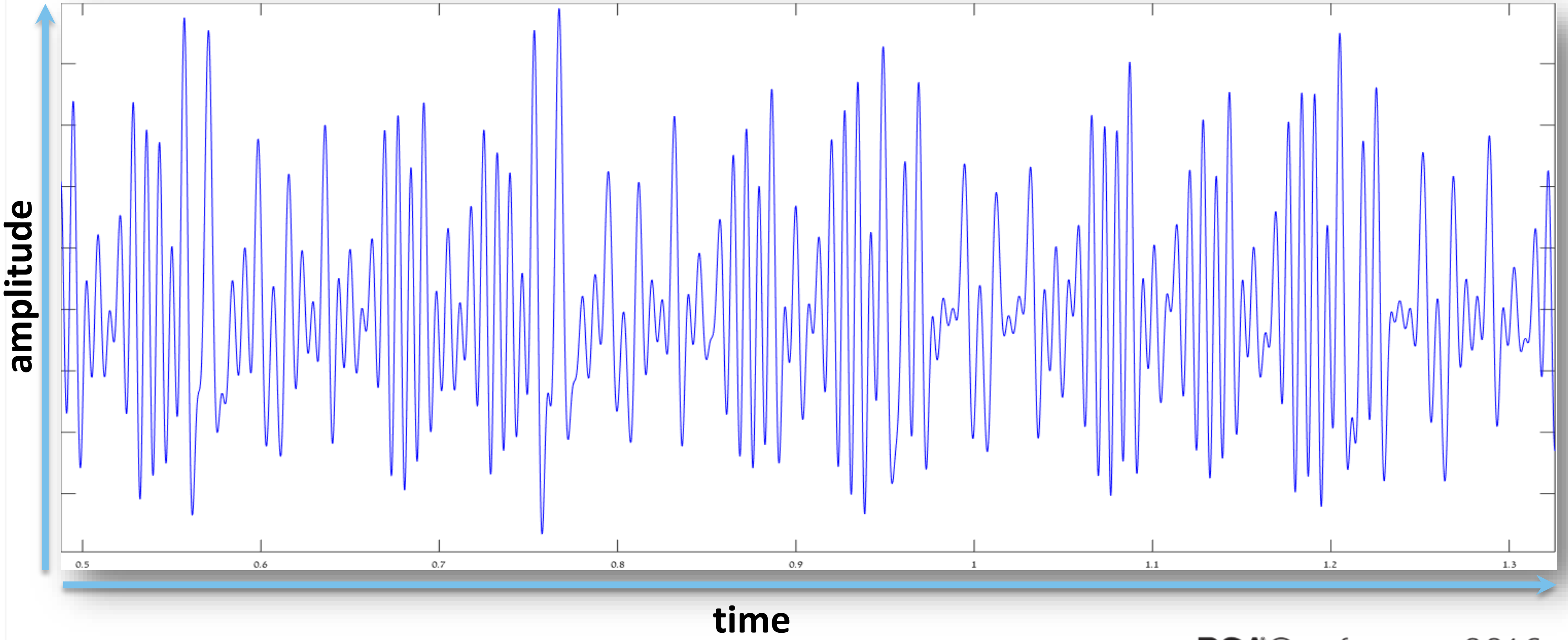


Obtained Signal



Empirical Results

Obtained Signal



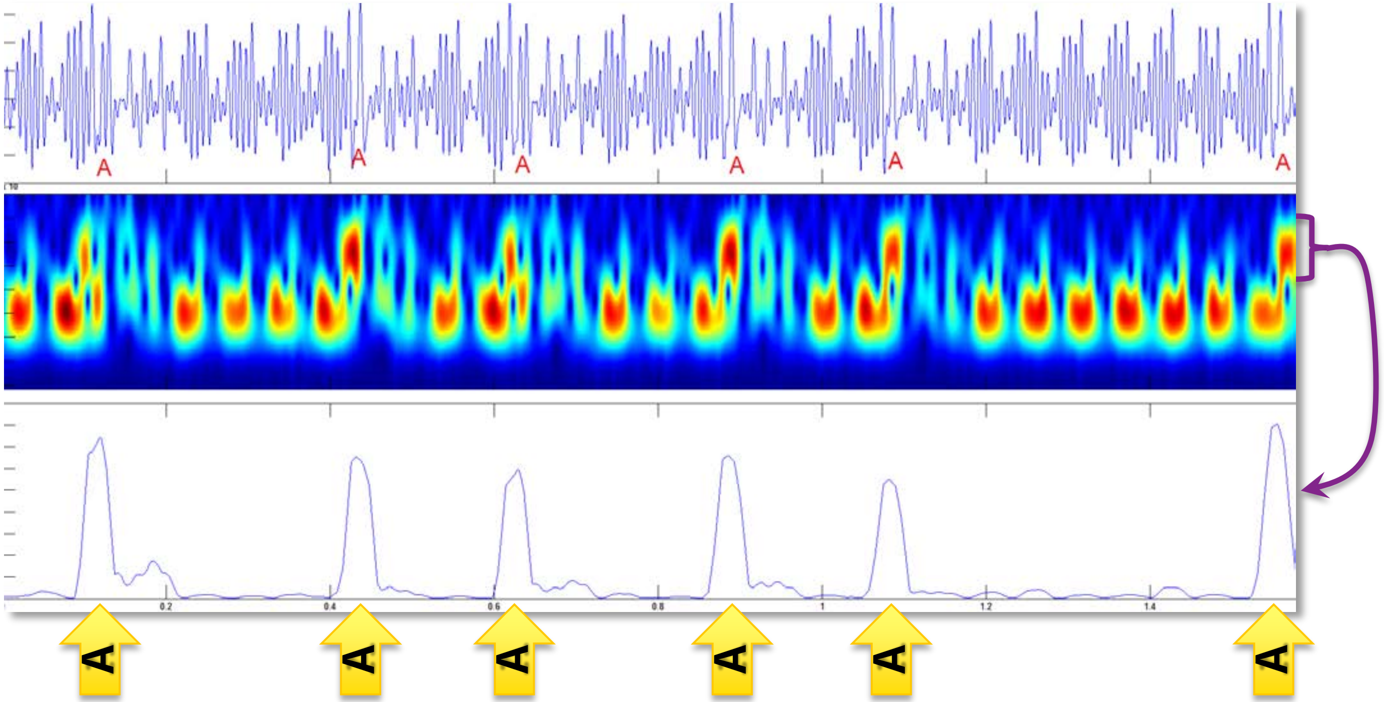
Distinguishing Add Operations

- Distinguishing between double and add operations

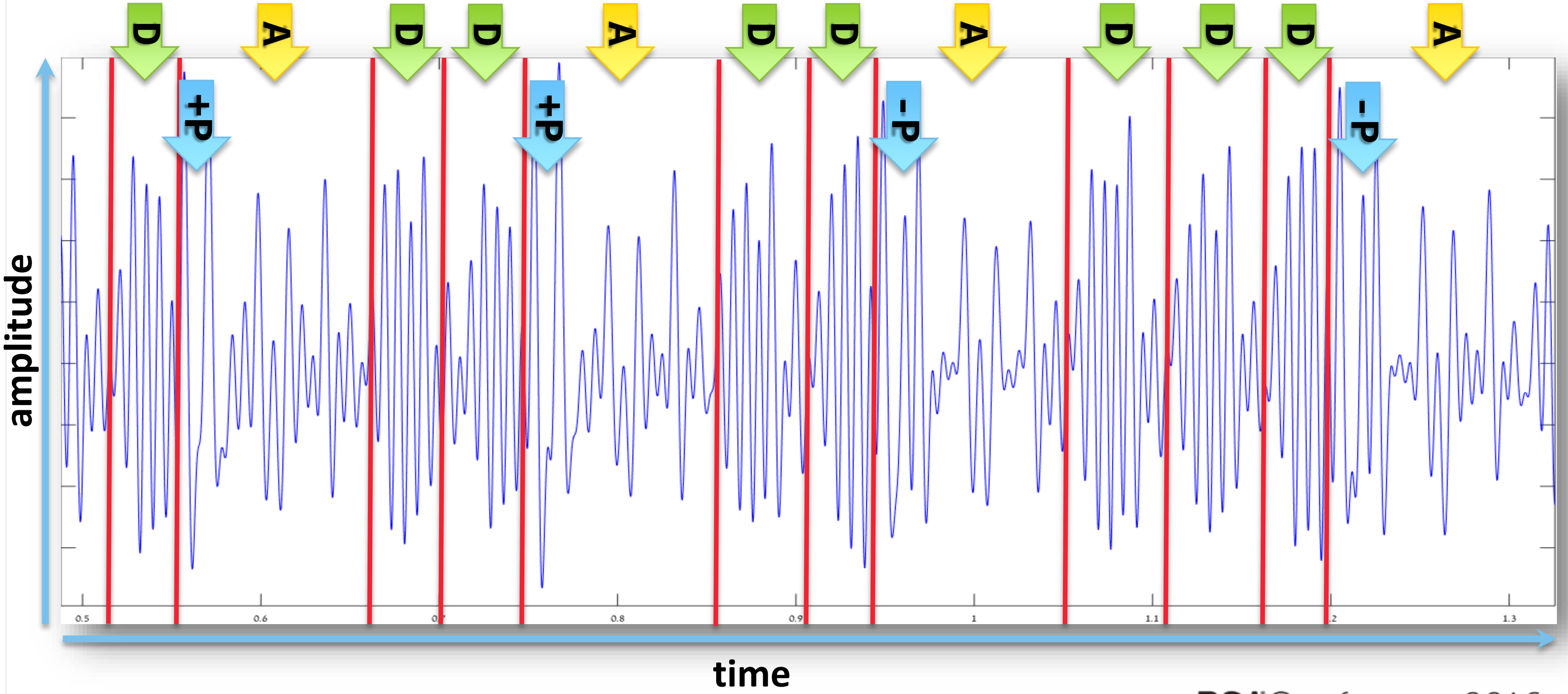
Aggregated Traces

Spectrogram

Energy of the higher frequency



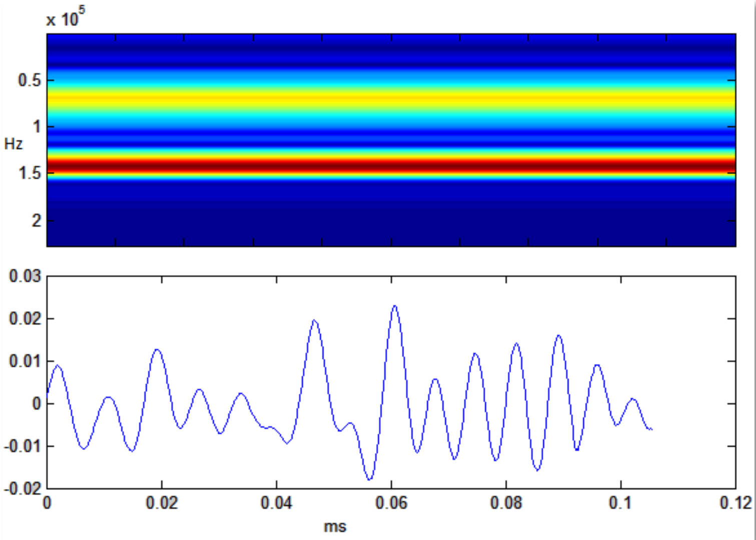
Obtained Signal



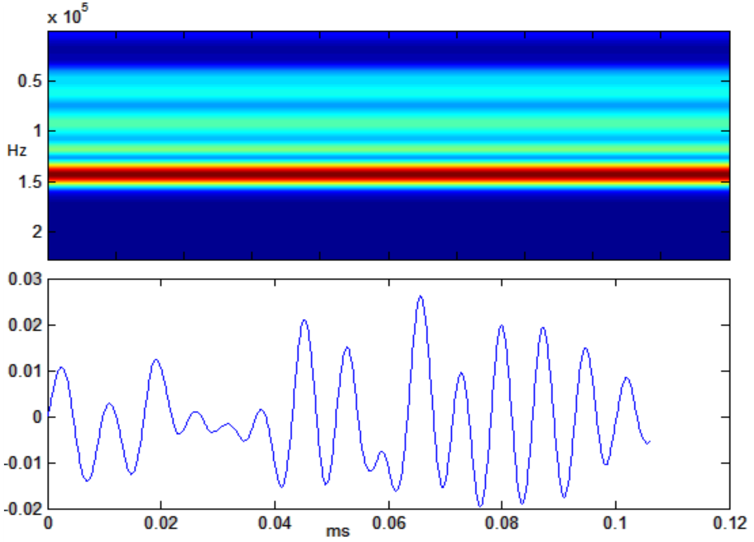
Distinguishing Between +1 and -1

- Using the timing information of add operations we zoom in

+1 NAF digit



-1 NAF digit



Conclusions and Countermeasures

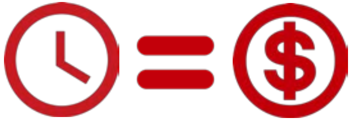
Overall ECDH attack

- Non-adaptive
  - 1 chosen ciphertext
- Low bandwidth
  - 5 MHz
- GHz scale PCs
  - Various models
- Fast
  - 66 decryptions
  - 3.3 seconds
- Common cryptographic software
  - GnuPG libgcrypt 1.6.3
  - CVE-2015-7511

Applying Countermeasures

- Change of scalar-by-point multiplication algorithm
  - Avoid key-dependent addition operations

- Scalar randomization
  - Split secret k to n parts $k = k_1 + \cdots + k_n$
  - Compute $k_1 \bullet \mathbb{P} + \cdots + k_n \bullet \mathbb{P}$
- Point blinding
  - Generate random point $\mathbb{R}$
  - Compute $k \bullet (\mathbb{P} + \mathbb{R}) - k \bullet \mathbb{R}$
- Careful constant-time, constant-cache implementation



Physical Side Channel Attacks on PCs

- Attacks are practical despite clock rates and noise
- Cheap, low-bandwidth attacks
- Applicable to common public-key algorithms
- Common software and hardware are vulnerable
- Many channels: EM, acoustic, power, ground-potential

Thanks!

cs.tau.ac.il/~tromer/ecdh

RSAConference2016

San Francisco | February 29 – March 4 | Moscone Center

Connect to Protect

SESSION ID: CRYP-R02

Side-Channel Attacks on Elliptic Curve Cryptography

Co-authors

Pierre Belgarric
Pierre-Alain Fouque
Gilles Macario-Rat
Mehdi Tibouchi

Pierre Belgarric

Research Engineer
HP Inc.

#RSAC

People

- Pierre Belgarric
  - PhD candidate at Orange Labs during this work
  - Now at HP Labs
  - Platform security
- Pierre-Alain Fouque
  - Université Rennes 1
  - Cryptanalyst
- Gilles Macario-Rat
  - Orange Labs
  - Cryptographer
- Mehdi Tibouchi
  - NTT, Japan
  - Cryptographer

Plan

- Introduction
- Evaluation environment
- Cryptanalysis of elliptic curves defined over prime fields
- Cryptanalysis of Koblitz curves
- Conclusion

Introduction

Context of the evaluation

- Sensitive services are being implemented on smartphones.
- Security challenges:
  - Security is built to protect against software vulnerabilities.
  - General-purpose hardware is not designed to be resistant to physical attacks.
- **Better evaluate the security of smartphones, and refine the threat model.**

Target specificities compared to smartcards

Hardware (physics)

- High-frequency clock
- Advanced semiconductor technology (in comparison to smartcards)
- Huge number of gates

45nm
65nm

Hardware (microarchitecture)

- Complex microarchitecture
- Multi-core
- Optimisation designs

ARMv7, Cortex A5
ARMv6, ARM11

Software

- Rich OS
- High number of threads
- Several stacks
- Applicative VM

Android
Dalvik VM

Related work

Early works

- Gebotys et al. (2005)
- Driss Aboulkassimi (2011)
- Kenworthy and Rohatgi (2012)

2014 – 2015: Main works

- Genkin et al. (x4)
- Longo et al.
- Balasch et al.

Evaluation environment

Evaluated software

- Study of Elliptic Curve Digital Signature Algorithm (ECDSA).
- Applicative library: Bouncy Castle.
- At the time of the study: version 1.50.
- in Dalvik as in Java, the library implementation is called through the JCA/JCE APIs.

Bouncy Castle Java library logo

Evaluated software

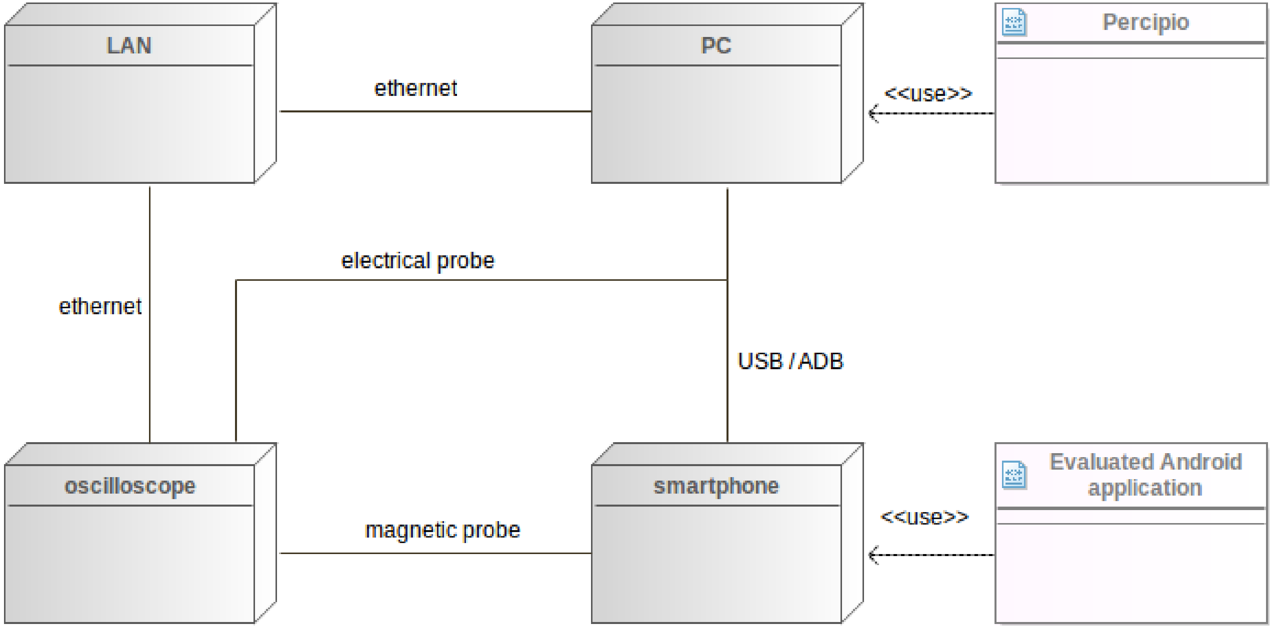
- Left-to-Right double and add wNAF algorithm
- Pre-computed points prevent from extracting value of added point with SPA

Algorithm 3 Left-to-Right double and add wNAF algorithm

Input: scalar k in wNAF $k_0, \ldots, k_n$ and precomputed points $\{P, \pm[3]P, \pm[5]P, \ldots, \pm[2^w - 1]P\}$
Output: *Point* $Q = kP$

```
1: function ScalarMultiplication(k, P)
2:     Q = ∞
3:     for i from n downto 0 do
4:         Q = 2 · Q
5:         if k_i ≠ 0 then Q = Q + [k_i]P
6:         end if
7:     end for
8:     return Q
9: end function
```

Experimental setup



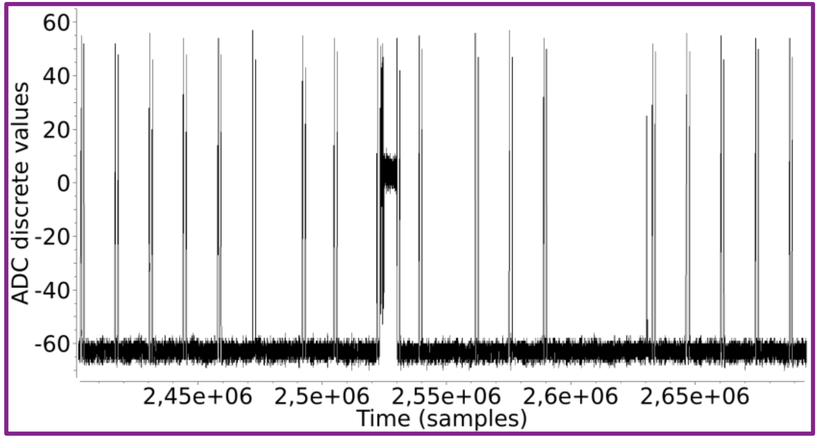
Side-channel evaluation bench

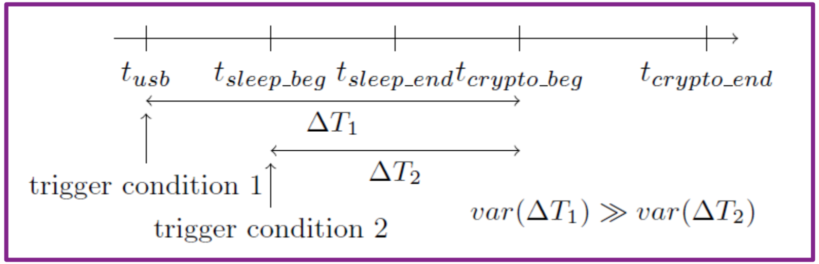
Experimental setup

- Observation of IC EM radiation.
- Near-field: magnetic loop probe within a few millimetres of the IC package
- Hundreds of measurements: automation required.
- Non-invasive: no tampering with the IC.

Synchronisation

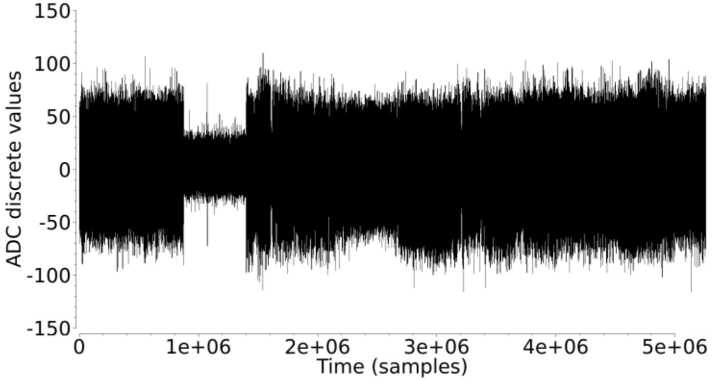
- PC sends signal to the smartphone on USB before encryption.
- Detected by oscilloscope.
- More accurate synchronisation using sleep instructions before cryptographic operations.



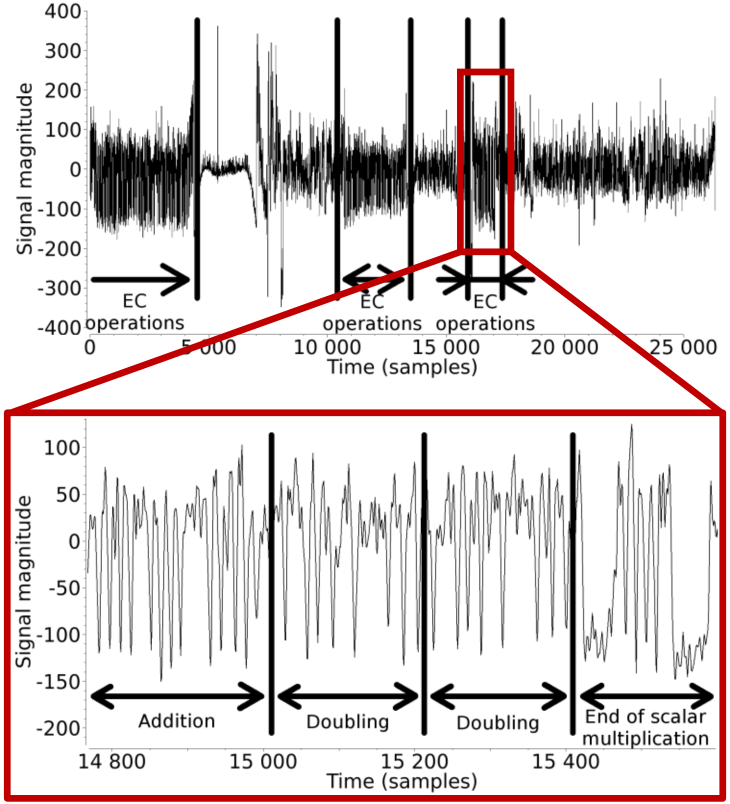


Cryptanalysis of elliptic curves defined over prime fields

Side-channel measurements



Digital signal filtering



Low-frequency leakages:

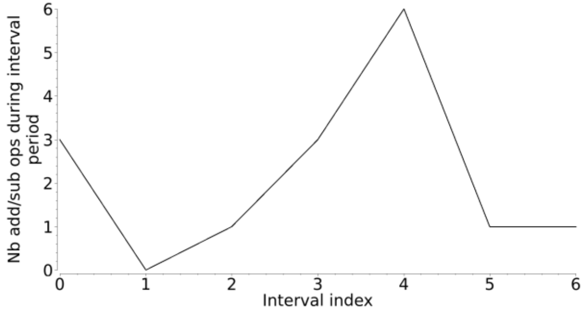
- signal is measured with 20 MHz low-pass filter
- a FIR filter is applied with 50 kHz cutting frequency
- CPU runs at 1.2 GHz

Leakage of the arithmetic multiplication

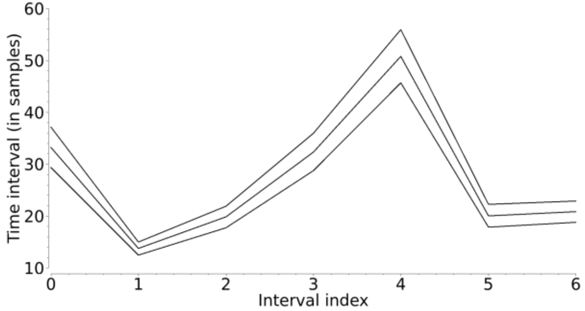
Algorithm 1 Doubling implementation in basic operations over Modified Jacobian coordinates in Bouncy Castle library

Input: *Point* $P_1 = (X_1, Y_1, Z_1, W_1)$ and boolean W
Output: *Point* $P_3 = (X_3, Y_3, Z_3, W_3)$

function ModifiedJacobianDoubling(W, P_1)
  $X1sq \leftarrow X_1 * X_1$
  $M \leftarrow ((X_1sq + X_1sq) + X_1sq) + W_1$
  $Y_1sq \leftarrow Y_1 * Y_1$
  $T \leftarrow Y_1sq * Y_1sq$
  $temp \leftarrow X_1 + Y_1sq$
  $temp_1 \leftarrow ((temp * temp) - X_1sq) - T$
  $S \leftarrow temp_1 + temp_1$
  $X_3 \leftarrow (M * M) - (S + S)$
  $temp_2 \leftarrow T + T$
  $temp_3 \leftarrow temp_2 + temp_2$
  $\_8T \leftarrow temp_3 + temp_3$
  $Y_3 \leftarrow (M * (S - X_3)) - \_8T$
  if $W = true$ **then**
    $temp_4 \leftarrow \_8T * W_1$
    $W_3 \leftarrow temp_4 + temp_4$
  end if
  if $Z_1.bitLen = 1$ **then**
    $temp_5 \leftarrow Y_1$
  else
    $temp_5 \leftarrow Y_1 * Z_1$
  end if
  $Z_3 \leftarrow temp_5 + temp_5$
  return $ECPoint.Fp(X_3, Y_3, Z_3, W_3)$
end function



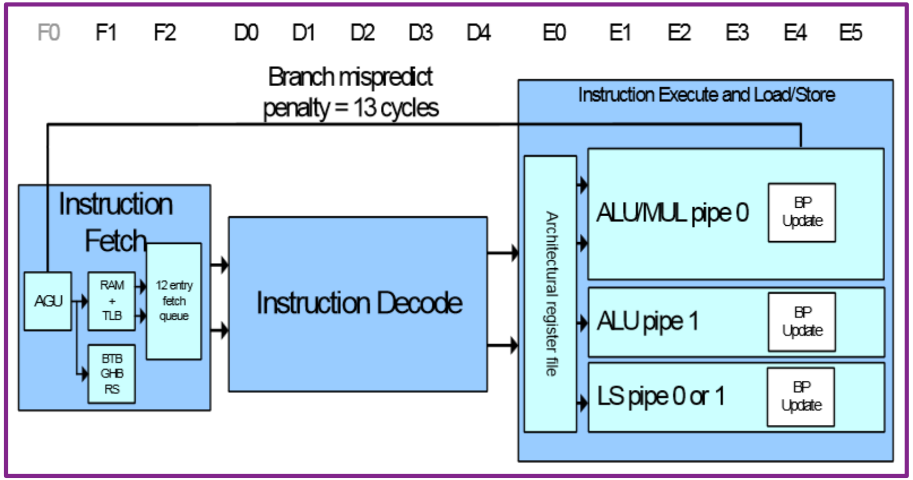
Number of basic operations between multiplications in double BC source code



Mean and standard deviation of doubling operation time intervals

Possible explanation

- Superscalar microarchitecture.
  - Multiple instructions run in parallel if possible.
  - Level of parallelism achievable depends on the program and the microarchitecture.
- Example of ARM Cortex-A8:
  - Arithmetic dual-pipeline.
  - Only one multiplier.
  - Might impact the number of execution pipelines in use.



A open question for further research: To what extent the microarchitecture impacts EM/power side-channels?

Lattice-based cryptanalysis on ECDSA

$$\begin{aligned} r(k) &= \lfloor x(kG) \rfloor_q \\ s(k,\mu) &= \left\lfloor k^{-1}\left(h(\mu) + \alpha r(k)\right) \right\rfloor_q \end{aligned}$$

ECDSA algorithm

$$k - a = 2^{\ell} b$$

a bits are known

$$\alpha r(k) 2^{-\ell} s(k,\mu)^{-1} \equiv \left(a - s(k,\mu)^{-1} h(\mu)\right) 2^{-\ell} + b \pmod q.$$

creating variables *t* and *u*:

$$\begin{aligned} t(k,\mu) &= \left\lfloor 2^{-\ell} r(k) s(k,\mu)^{-1} \right\rfloor_q \\ u(k,\mu) &= \left\lfloor 2^{-\ell}\left(a - s(k,\mu)^{-1} h(\mu)\right) \right\rfloor_q \end{aligned}$$

knowledge of close value

$$0 \le \lfloor \alpha t(k,\mu) - u(k,\mu) \rfloor_q < q/2^{\ell}$$

$$|\alpha t(k,\mu) - u(k,\mu) - q/2^{\ell+1}|_q \le q/2^{\ell+1}$$

several signatures

$$\begin{pmatrix} q & 0 & \cdots & 0 & 0 \\ 0 & q & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \dots & 0 & q & 0 \\ t_1 & \dots & \dots & t_d & 1/2^{\ell+1} \end{pmatrix}$$

~ Hidden Number Problem HNP

Able to extract the key using a little more of 500 signatures

Cryptanalysis of Koblitz curves

Koblitz curves

- Efficient implementation in hardware and in software
- Anomalous curves defined with an equation of the form:

$$E_a(\mathbb{F}_{2^m}): \qquad y^2 + xy = x^3 + ax + 1, \text{ and } \; a = 0 \text{ or } 1.$$

- Frobenius map:

$$\tau : E_a(\mathbb{F}_{2^m}) \to E_a(\mathbb{F}_{2^m}) \qquad \tau(\infty) = \infty, \text{ and } \tau(x,y) = (x^2, y^2).$$

Koblitz curves

- The points of the curve satisfy the equation:

$$(\tau^2 + 2)P = \mu\tau(P) \text{ for all } P \in E_a(\mathbb{F}_{2^m}), \quad \mu = (-1)^a$$

- The Frobenius map can be seen as the complex number:

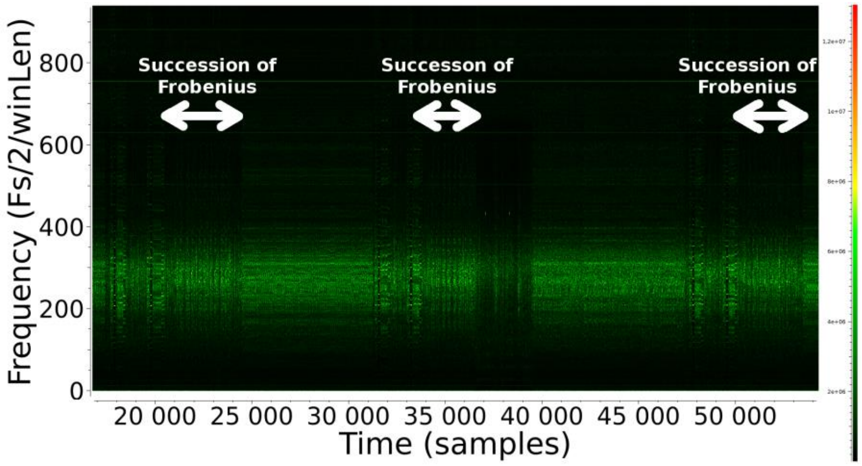
$$\tau = (\mu + \sqrt{-7})/2$$

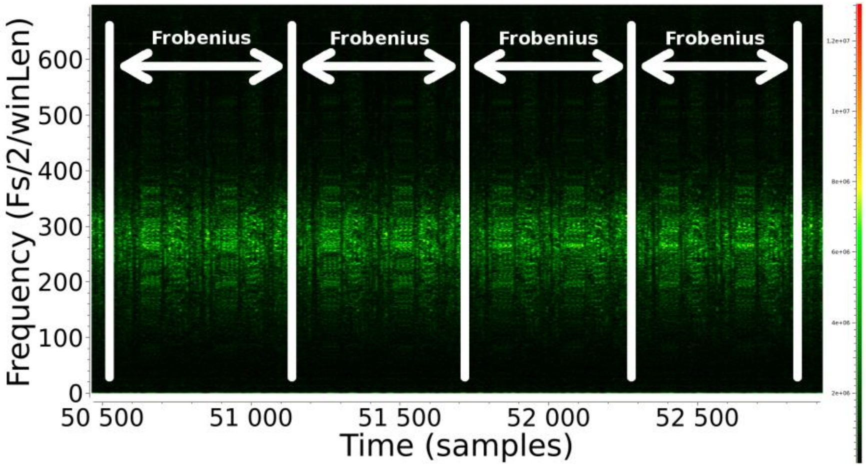
- Representing the scalar k in a tau-adic base, then doubling is a Frobenius:

$$u_{l-1}\tau^{l-1} + \cdots + u_1\tau + u_0$$

Observed leakage

- Frobenius operation is very performant
- Pre-computed tables in Bouncy Castle
- Short-Term Fourier Transform (STFT)





New Cryptanalysis

- Extension of the classical HNP attack on ECDSA using lattice reduction
- Works by representing scalars in the form $a_0 + a_1\tau$ with a0, a1 half-size integers
- The magic that makes things tick is the fact that $|\tau| = \sqrt{2}$
- The overall extension is not very hard, but the precise analysis of the extended attack is surprisingly subtle
- Upshot: the bias/leakage needed to mount an attack for a certain field size is larger than in the classical case, but not by a large margin (only a fraction of a bit for random TNAFs)

Conclusion

Potential Use Case: Bitcoin

- Bitcoin wallets
  - A wallet is a pair of EC private key.
  - The elliptic curve is Secp256k1.
- Android wallets
  - Android Bitcoin wallets usually rely on Bitcoinj.
  - Bitcoinj is built upon Spongycastle for cryptography.
  - Spongycastle is a library adaptation of Bouncy Castle for Android.
- Our cryptanalysis of curves defined over prime fields could be used to extract key from a wallet spending money.
- Still some challenges to become a real-world threat:
  - Hundreds of Bitcoin payments to observe,
  - Near-field EM radiation,
  - Synchronisation on USB cable.

Branch: master ▾ bitcoinj / core / src / main / java / org / bitcoinj / core / **ECKey.java**

schildbach Always print to the log, rather than to the console.

7 contributors

1262 lines (1143 sloc) 57.2 KB

```
/*
 * Copyright 2011 Google Inc.
 * Copyright 2014 Andreas Schildbach
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.bitcoinj.core;

import org.bitcoinj.crypto.*;
import com.google.common.annotations.VisibleForTesting;
import com.google.common.base.MoreObjects;
import com.google.common.base.Objects;
import com.google.common.base.Preconditions;
import com.google.common.primitives.Ints;
import com.google.common.primitives.UnsignedBytes;
import org.bitcoin.NativeSecp256k1;
import org.bitcoinj.wallet.Protos;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.spongycastle.asn1.*;
import org.spongycastle.asn1.x9.X9ECParameters;
import org.spongycastle.asn1.x9.X9IntegerConverter;
import org.spongycastle.crypto.AsymmetricCipherKeyPair;
import org.spongycastle.crypto.digests.SHA256Digest;
import org.spongycastle.crypto.ec.CustomNamedCurves;
import org.spongycastle.crypto.generators.ECKeyPairGenerator;
import org.spongycastle.crypto.params.*;
import org.spongycastle.crypto.signers.ECDSASigner;
import org.spongycastle.crypto.signers.HMacDSAKCalculator;
import org.spongycastle.math.ec.ECAlgorithms;
import org.spongycastle.math.ec.ECPoint;
import org.spongycastle.math.ec.FixedPointCombMultiplier;
import org.spongycastle.math.ec.FixedPointUtil;
import org.spongycastle.math.ec.custom.sec.SecP256K1Curve;
import org.spongycastle.util.encoders.Base64;
```

Conclusion / Perspectives

- Hardware physical attack surface must be considered more often.
- Root causes of the leakage observed are not fully understood yet.
  - In particular, how the microarchitecture impacts EM/power side-channels.
- No individual system component was faulty:
  - General purpose SoCs are not specified to protect against physical attacks.
  - The crypto library was not expected to protect against physical attacks.
- Suitable counter-measures should be implemented at algorithmic / software levels.
- Recent Bouncy Castle protects against the attack presented here: implementing scalar multiplication with the Fixed-point Comb algorithm.

Apply

- **Threats**: Consider that physical side-channel is a realistic threat.
- **Developers**: Check that implementation is secure against physical attacks.
- **Researchers**: Go further into the root causes of vulnerabilities.