ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs
## Key Extraction via Physical Side Channels

### Small Devices
- Modular Exponentiation (RSA, ElGamal)
- Elliptic Curve Cryptography

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

### Big Devices
- [Fouque Kunz-Jacques Martinet Müller Valette 06] [Gandolfi Mortel Oliver 01]
- [Homma Miyamoto Aoki Satoh Shamir 08]
- [Kocher 96] [Courrege Feix Roussellet 10]
- [Fouque Valette 03] [Kocher Jaffe Jum 99]
- [Messerges Dabbish Sloan 99] [Novak 02]
- [Walter Thompson 01] [Kühn 03]...

### 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)

### New Challenges
- Shorter keys, smaller numbers - even faster
- Different math

### This Paper
- [Genkin Shamir Tromer 14]
- [Genkin Pipman Tromer 14]
- [Genkin Pachmanov Pipman Tromer 15]

---

### This Paper

- Acoustic
- EM, ground potential
- Cheap EM

---

### Modular Exponentiation (RSA, ElGamal)

- [Cron 02], [Akishita Takagi 03], [Avanzi 05], [Biehl Meyer Müller 00], [Blömer Otto Seifert 06], [Ciet Joyce 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]...
Attacking ECDH: GnuPG as a case study
**Elliptic Curve Diffie-Hellman (ECDH) Encryption**

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

- **Encryption:**
  - Random number: $k'$
  - Ephemeral key: $t = KDF(k' \cdot (k \cdot G))$
  - Ciphertext: $c = (AES_t(m), k' \cdot G)$

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

- **Standardized**
  - OpenPGP [RFC 6637]
  - NIST SP800-56A

- **Implementations**
  - GnuPG (libgcrypt)
  - BouncyCastle
  - Google’s end-to-end encrypted email
GnuPG’s NAF representation

- Non-Adjacent Form (NAF) representation [Reitwiesner 60]
  - Allows positive and negative digits
  - \( b = \sum_i 2^i b_i \) where \( b_i \in \{-1, 0, 1\} \)
  - Reduces the number of nonzero digits from \( \frac{1}{2} \) to \( \frac{1}{3} \)
  - Example: \( 7 = (0, 1, 1, 1)_2 = (1, 0, 0, -1)_2 \)
GnuPG’s Scalar-by-Point Multiplication

\[
\text{point\_mul}(k, P) \{
A = P
\text{for}\ i = n-1..0\ \text{do}
A = 2 \cdot A
\text{if}\ k[i] == 1\ \text{then}
A = A + P
\text{if}\ k[i] == -1\ \text{then}
P' = -P
A = A + P'
\text{return}\ A
\}
\]

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

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

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

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

\[
A = [k_n \| \ldots \| k_{i+1} \| k_i] \cdot P
\]
GnuPG’s Scalar-by-Point Multiplication

```c
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
}
```

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

5MHz measurements vs. 2000MHz CPU

$k = 1, 0, -1, -1, \ldots$
GnuPG’s Scalar-by-Point Multiplication

```c
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
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
}

point_add(P1, P2) {
    if P1.z==0 then return P2
    if P2.z==0 then return P1
    t1 = P1.x*(P2.z^2)
    t2 = P2.x*(P1.z^2)
    t3 = t1-t2
    t4 = P1.y*(P2.z^3)
    t5 = P2.y*(P1.z^3)
    if t3==0 && t6==0 then return (1,1,0)
    P3.x = t6^2-t7*t3^2
    t9 = t7*t3^2-2*P3.x
    P3.y = (t9*t6-t8*t3^3)/2
    P3.z = z1*z2*t3
    return P3
}
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

- Amplitude
- Time

Symbols:
- D
- A
- P

Note: RSA Conference 2016 watermark is present.
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 $k_1 \cdot \mathbb{P} + \cdots + k_n \cdot \mathbb{P}$

- Point blinding
  - Generate random point $\mathbb{R}$
  - Compute $k \cdot (\mathbb{P} + \mathbb{R}) - k \cdot \mathbb{R}$

- 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
Side-Channel Attacks on Elliptic Curve Cryptography
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
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.
- 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:** $Q = kP$

1. function `SCALAR_MULTIPLICATION(k, P)`
2. $Q = \infty$
3. for $i$ from $n$ downto $0$ do
4. \hspace{1em} $Q = 2 \cdot Q$
5. \hspace{1em} if $k_i \neq 0$ then $Q = Q + [k_i]P$
6. \hspace{1em} 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
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 $P_2 = (X_3, Y_3, Z_3, W_3)$

1: function MODIFIED_JACOBIAN_DOUBLING($W, P_1$)
2: $X_{1sq} ← X_1 * X_1$
3: $M ← ((X_{1sq} + X_1) + X_{1sq}) + W_1$
4: $Y_{1sq} ← Y_1 * Y_1$
5: $T ← Y_{1sq} * Y_{1sq}$
6: $\text{temp} ← X_1 + Y_1 sq$
7: $\text{temp}_1 ← \text{((temp} * \text{temp}) - X_{1sq} - T$
8: $S ← \text{temp}_1 + \text{temp}_1$
9: $X_3 ← (M * M) - (S + S)$
10: $\text{temp}_2 ← T + T$
11: $\text{temp}_3 ← \text{temp}_2 + \text{temp}_2$
12: $\_ST ← \text{temp}_3 + \text{temp}_3$
13: $Y_3 ← (M * (S - X_3)) - \_ST$
14: if $W = \text{true}$ then
15: $\text{temp}_4 ← \_ST * W_1$
16: $W_3 ← \text{temp}_4 + \text{temp}_4$
17: end if
18: if $Z_1 \_bitLen = 1$ then
19: $\text{temp}_5 ← Y_1$
20: else
21: $\text{temp}_5 ← Y_1 * Z_1$
22: end if
23: $Z_3 ← \text{temp}_5 + \text{temp}_5$
24: return $ECPoint.Fp(X_3, Y_3, Z_3, W_3)$
25: 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

\[ r(k) = [x(kG)]_q \]
\[ s(k, \mu) = [k^{-1}(h(\mu) + \alpha r(k))]_q \]

ECDSA algorithm

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

Creating variables \( t \) and \( u \):

\[ t(k, \mu) = [2^{-\ell}r(k)s(k, \mu)^{-1}]_q \]
\[ u(k, \mu) = [2^{-\ell}(a - s(k, \mu)^{-1}h(\mu))]_q \]

Knowledge of close value

0 ≤ |\( \alpha t(k, \mu) - u(k, \mu) \)| < \( q/2^{\ell} \)

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

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:

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

- Frobenius map:

  \[ \tau : E_a(\mathbb{F}_{2^m}) \to E_a(\mathbb{F}_{2^m}) \quad \tau(\infty) = \infty, \text{ and } \tau(x, y) = (x^2, y^2). \]
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 $a_0$, $a_1$ 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.
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.