ECDSA White-Box Implementations
Feedback on CHES 2021 WhibOx Contest

Agathe Houzelot

October 18, 2023
Black-Box, Grey-Box, White-Box

Plaintext

Ciphertext

Cryptanalysis

Plaintext

Ciphertext

Side-channels/Faults

Plaintext

Ciphertext

Read/modify the binary/memory
### Look-up tables and encodings

<table>
<thead>
<tr>
<th>input</th>
<th>0</th>
<th>1</th>
<th>...</th>
<th>$2^{128} - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>$AES_k(0)$</td>
<td>$AES_k(1)$</td>
<td>...</td>
<td>$AES_k(2^{128} - 1)$</td>
</tr>
</tbody>
</table>
### Look-up tables and encodings

<table>
<thead>
<tr>
<th>input</th>
<th>0</th>
<th>1</th>
<th>…</th>
<th>$2^{128} - 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>$\text{AES}_k(0)$</td>
<td>$\text{AES}_k(1)$</td>
<td>…</td>
<td>$\text{AES}_k(2^{128} - 1)$</td>
</tr>
</tbody>
</table>

![Diagram](image)
State of the Art

- **2002**: 1st AES: white-box
- **2004**: BGE attack
- **2006, 2009, 2010**: New propositions for AES
- **2010, 2012, 2013**: New attacks
- **2016**: DCA: white-box side-channels
- **2017, 2019, 2021**: WhibOx Contests (AES, AES, ECDSA)
- **2020**: 1st ECDSA white-box

- New designs
- New attacks
- Both designs and attacks
CHES 2021 Challenge - the WhibOx Contest

Designers
- Post C codes computing ECDSA
- Challenges gain strawberries (depending on performances and time until break)

Attackers
- Try to extract the secret key
- Receive bananas (number of strawberries of the challenge)
Our Contributions [1]

**zerokey**
- Posted the 2 winning challenges
- Described the implementations

**TheRealIdefix**
- Broke the most challenges
- Described attacks, showing which ones succeeded for each candidate
Let \( G \) be a point of order \( n \) on an elliptic curve \( E \)

Let \( d \) be a 256-bit key

Let \( m \) be a message and \( e = H(m) \) its hash value

**Algorithm 1: ECDSA signature**

1. \( k \leftarrow [1, n - 1] \)
2. \( R \leftarrow kG \)
3. \( r \leftarrow x_R \mod n \)
4. \( s \leftarrow k^{-1}(e + rd) \mod n \)
5. **if** \( r == 0 \) **or** \( s == 0 \) **then**
   6. Go to step 1
6. **end**
7. Return \((r,s)\)
Let $G$ be a point of order $n$ on an elliptic curve $E$
Let $d$ be a 256-bit key
Let $m$ be a message and $e = H(m)$ its hash value

**Algorithm 1: ECDSA signature**

1. $k \leftarrow [1, n - 1]$
2. $R \leftarrow kG$
3. $r \leftarrow x_R \mod n$
4. $s \leftarrow k^{-1}(e + rd) \mod n$
5. **if** $r == 0$ or $s == 0$ **then**
6. | Go to step 1
7. **end**
8. Return $(r, s)$
ECDSA Sensitive Values

- Let $G$ be a point of order $n$ on an elliptic curve $E$
- Let $d$ be a 256-bit key
- Let $m$ be a message and $e = H(m)$ its hash value

Algorithm 1: ECDSA signature

1. $k \leftarrow \mathbb{Z}_{[1, n - 1]}$
2. $R \leftarrow kG$
3. $r \leftarrow x_R \mod n$
4. $s \leftarrow k^{-1}(e + rd) \mod n$
5. if $r == 0$ or $s == 0$ then
   6.   Go to step 1
7. end
8. Return $(r, s)$
Deterministic ECDSA

- Let $G$ be a point of order $n$ on an elliptic curve $E$
- Let $d$ be a 256-bit key
- Let $m$ be a message and $e = H(m)$ its hash value

**Algorithm 1: ECDSA signature**

```plaintext
1. $k \leftarrow [1, n − 1]$          WB model $\Rightarrow$ No reliable source of randomness!
2. $R \leftarrow kG$
3. $r \leftarrow x_R \mod n$
4. $s \leftarrow k^{-1}(e + rd) \mod n$
5. **if** $r == 0$ **or** $s == 0$ **then**
   6.     Go to step 1
5. **end**
8. Return $(r, s)$
```
Deterministic ECDSA

- Let $G$ be a point of order $n$ on an elliptic curve $E$
- Let $d$ be a 256-bit key
- Let $m$ be a message and $e = H(m)$ its hash value

**Algorithm 1: ECDSA signature**

1. $k \leftarrow f(e)$ \quad \text{WB model} \Rightarrow \text{No reliable source of randomness!}
2. $R \leftarrow kG$
3. $r \leftarrow x_R \mod n$
4. $s \leftarrow k^{-1}(e + rd) \mod n$
5. **if** $r == 0$ **or** $s == 0$ **then**
   6. \quad \text{Go to step 1}
5. **end**
7. Return $(r, s)$
Memory dumps

Idea

Find some secret values that could be manipulated in the clear

- Easy since we had access to a C code and not a binary
- Usual encoding techniques not suited for operations on big numbers → one has to design new techniques
First possibility

Find collisions: signatures generated with the same nonce

- Find \((r_1, s_1)\) and \((r_2, s_2)\) such that \(r_1 = r_2\) (so \(k_1 = k_2\))
- Solve the following system in \(k, d\):

\[
\begin{align*}
    s_1 &= k^{-1}(e_1 + r_1d) \\
    s_2 &= k^{-1}(e_2 + r_2d)
\end{align*}
\]
Biased Nonce

First possibility
Find collisions: signatures generated with the same nonce

- Find \((r_1, s_1)\) and \((r_2, s_2)\) such that \(r_1 = r_2\) (so \(k_1 = k_2\))
- Solve the following system in \(k, d\):

\[
\begin{aligned}
  s_1 &= k^{-1}(e_1 + r_1d) \\
  s_2 &= k^{-1}(e_2 + r_2d)
\end{aligned}
\]

Second possibility
Exploit biases in the nonce generation

- Use lattice-based attacks
- Allows to recover \(d\) from a few bits of \(k\) for several signatures
Grey-Box Attacks in the White-Box Model

- Side-channel attacks
  - Difficult to apply (huge size of the traces)

Fault injections
- Modify the binary, use debugging tools
  - very precise faults
- Many fault attacks on deterministic ECDSA, for example:

Valid signature

\[ r = x R \mod n \]
\[ s = k - 1 (e + rd) \mod n \]

Faulty signature

\[ r' = x R \mod n \]
\[ s' = k - 1 (e + r'd) \mod n \]

\[ d = (s (r - r') (s - s') - 1 - r) - 1 e \mod n \]
Grey-Box Attacks in the White-Box Model

- Side-channel attacks
  - Difficult to apply (huge size of the traces)

- Fault injections
  - Modify the binary, use debugging tools ⇒ very precise faults
  - Many fault attacks on deterministic ECDSA, for example:

  Valid signature
  \[ r = x_R \mod n \]
  \[ s = k^{-1}(e + rd) \mod n \]
Grey-Box Attacks in the White-Box Model

- Side-channel attacks
  - Difficult to apply (huge size of the traces)

- Fault injections
  - Modify the binary, use debugging tools ⇒ very precise faults
  - Many fault attacks on deterministic ECDSA, for example:

Valid signature

\[ r = x_R \pmod{n} \]
\[ s = k^{-1}(e + rd) \pmod{n} \]

Faulty signature

\[ r' = x_{R'} \pmod{n} \]
\[ s' = k^{-1}(e + r'd) \pmod{n} \]
Grey-Box Attacks in the White-Box Model

- Side-channel attacks
  - Difficult to apply (huge size of the traces)

- Fault injections
  - Modify the binary, use debugging tools ⇒ very precise faults
  - Many fault attacks on deterministic ECDSA, for example:

\[
\begin{align*}
\text{Valid signature} & \quad \text{Faulty signature} \\
 r &= x_R \mod n & r' &= x_{R'} \mod n \\
 s &= k^{-1}(e + rd) \mod n & s' &= k^{-1}(e + r'd) \mod n \\
 d &= (s(r - r')(s - s')^{-1} - r)^{-1}e \mod n &
\end{align*}
\]
Percentage of Challenges Broken by Each Attack

- Bad nonce: 74%
- Faults: 78%
- Memory dump: 33%
Conclusion

- Securing ECDSA seems even more difficult than the AES
  - Our automated attacks broke 95 out of 97 challenges
  - All the challenges were broken in less than 33 hours

What about the ECDSA white-box published in 2020?
- Broken too but with a more sophisticated fault attack [2]

Is there any hope?
- Possible to increase a lot the workload of the attacker
- Companies sell ECDSA white-boxes evaluated by specialized labs and not broken

Agathe Houzelot
Conclusion

- Securing ECDSA seems even more difficult than the AES
  - Our automated attacks broke 95 out of 97 challenges
  - All the challenges were broken in less than 33 hours
- What about the ECDSA white-box published in 2020?
  - Broken too but with a more sophisticated fault attack [2]
Conclusion

- Securing ECDSA seems even more difficult than the AES
  - Our automated attacks broke 95 out of 97 challenges
  - All the challenges were broken in less than 33 hours
- What about the ECDSA white-box published in 2020?
  - Broken too but with a more sophisticated fault attack [2]
- Is there any hope?
  - Possible to increase a lot the workload of the attacker
  - Companies sell ECDSA white-boxes evaluated by specialized labs and not broken
Thank you for your attention!

Any question?
Ecdsa white-box implementations: Attacks and designs from ches 2021 challenge.

C. Giraud and A. Houzelot.
Fault attacks on a cloud-assisted ecdsa white-box based on the residue number system.