CHES 2017
Capture the Flag Challenge
The WhibOx Contest
An ECRYPT White-Box Cryptography Competition
Organisation

• Organised by the ECRYPT-CSA project

• Submission server developed by CryptoExperts

• Submission server hosted by TU Eindhoven
White-box crypto

- Obfuscation for crypto implementations
- Should at least be secure against key extraction
- Every published scheme broken
- Big trend in the industry (mobile payment, DRM, …)
- Deployed implementations based on secret technologies

(picture source: http://www.whiteboxcrypto.com/)
White-box contest

- Following an open discussion at the WhibOx workshop (co-located with CHES & CRYPTO 2016)

- Goal: confront designers and attackers of practical white-box crypto

- Designers can submit WB AES implementations st:
  - C source code at most 50MB
  - Executable at most 20MB
  - Use at most 20MB of RAM
  - Run in at most 1 second

- Attackers can try to recover the keys of submitted implem.
Contest rules

• A WB implem. gets 🍓 points as long as it stays unbroken

• $n$ new 🍓 on day $n$ (quadratic growth)

• When a WB implem. with $q$ 🍓 is broken:
  • the attacker gets $q$ 🍌 points (with max rule)
  • the 🍓 score of the implem. starts to decrease symmetrically down to 0
  • the designer of the implem. gets $q$ 🍓 points
Strawberry scores over time
No implementation got more than 1 strawberry before 08/20.
Strawberry scores over time

Everything was broken in the end!

No implementation got more than 1 🍓 before 08/20
Strawberry scores over time

Everything was broken in the end!

No implementation got more than 1 strawberry before 08/20

Outstanding winner
Strawberry scores over time

No implementation got more than 1 🍓 before 08/20

Everything was broken in the end!

Outstanding winner

Several challenging implementations
Submissions over time

Server upgrade

Server down
Number of breaks over time
Slack activity

20772 messages + 615 files exchanged
Challenges were broken 9.33 times on average

<table>
<thead>
<tr>
<th>#Breaks</th>
<th>Most broken</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>wizardly_shannon</td>
</tr>
<tr>
<td>37</td>
<td>angry_meitner</td>
</tr>
<tr>
<td>20</td>
<td>hopeful_liskov</td>
</tr>
<tr>
<td></td>
<td>quirky_keller</td>
</tr>
<tr>
<td>18</td>
<td>elegant_sinoussi</td>
</tr>
<tr>
<td>16</td>
<td>stupefied_varahamihira</td>
</tr>
<tr>
<td>14</td>
<td>famous_stonebraker</td>
</tr>
<tr>
<td></td>
<td>practical_cori</td>
</tr>
<tr>
<td>13</td>
<td>eloquent_indiana</td>
</tr>
<tr>
<td></td>
<td>festive_jennings</td>
</tr>
<tr>
<td></td>
<td>modest_clarke</td>
</tr>
<tr>
<td></td>
<td>zealous_ardinghelli</td>
</tr>
<tr>
<td></td>
<td>determined_goldwasser</td>
</tr>
<tr>
<td></td>
<td>nostalgic_noether</td>
</tr>
<tr>
<td></td>
<td>vigilant_heyrovsky</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#Breaks</th>
<th>Least broken</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>mystifying_galileo</td>
</tr>
<tr>
<td></td>
<td>silly_feynman</td>
</tr>
<tr>
<td></td>
<td>priceless_stallman</td>
</tr>
<tr>
<td></td>
<td>relaxed_alen</td>
</tr>
<tr>
<td></td>
<td>musing_lalande</td>
</tr>
<tr>
<td></td>
<td>compassionate_albattani</td>
</tr>
<tr>
<td></td>
<td>smart_ardinghelli</td>
</tr>
<tr>
<td></td>
<td>angry_jones</td>
</tr>
<tr>
<td></td>
<td>nervous_montalcini</td>
</tr>
<tr>
<td>3</td>
<td>sad_goldstine</td>
</tr>
<tr>
<td></td>
<td>bright_morse</td>
</tr>
<tr>
<td>2</td>
<td>relaxed_brown</td>
</tr>
<tr>
<td></td>
<td>hungry_clarke</td>
</tr>
<tr>
<td></td>
<td>vibrant_goldberg</td>
</tr>
<tr>
<td>1</td>
<td>jolly_davinci</td>
</tr>
<tr>
<td></td>
<td>competent_agnesi</td>
</tr>
<tr>
<td></td>
<td>adoring_poitras</td>
</tr>
</tbody>
</table>
Challenges were broken 9.33 times on average

<table>
<thead>
<tr>
<th>#Breaks</th>
<th>Most broken</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>wizardly_shannon</td>
</tr>
<tr>
<td>37</td>
<td>angry_meitner</td>
</tr>
<tr>
<td>20</td>
<td>hopeful_liskov</td>
</tr>
<tr>
<td></td>
<td>quirky_keller</td>
</tr>
<tr>
<td>18</td>
<td>elegant_sinoussi</td>
</tr>
<tr>
<td>16</td>
<td>stupefied_varahamihira</td>
</tr>
<tr>
<td>14</td>
<td>famous_stonebraker</td>
</tr>
<tr>
<td></td>
<td>practical_cori</td>
</tr>
<tr>
<td>13</td>
<td>eloquent_indiana</td>
</tr>
<tr>
<td>12</td>
<td>festive_jennings</td>
</tr>
<tr>
<td></td>
<td>modest_clarke</td>
</tr>
<tr>
<td></td>
<td>zealous_ardinghelli</td>
</tr>
<tr>
<td></td>
<td>determined_goldwasser</td>
</tr>
<tr>
<td></td>
<td>nostalgic_noether</td>
</tr>
<tr>
<td></td>
<td>vigilant_heyrovsky</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#Breaks</th>
<th>Least broken</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>mystifying_galileo</td>
</tr>
<tr>
<td></td>
<td>silly_feynman</td>
</tr>
<tr>
<td></td>
<td>priceless_stallman</td>
</tr>
<tr>
<td></td>
<td>relaxed_allen</td>
</tr>
<tr>
<td></td>
<td>musing_lalande</td>
</tr>
<tr>
<td></td>
<td>compassionate_albattani</td>
</tr>
<tr>
<td></td>
<td>smart_ardinghelli</td>
</tr>
<tr>
<td></td>
<td>angry_jones</td>
</tr>
<tr>
<td></td>
<td>relaxed_allen</td>
</tr>
<tr>
<td></td>
<td>musing_lalande</td>
</tr>
<tr>
<td></td>
<td>compassionate_albattani</td>
</tr>
<tr>
<td></td>
<td>smart_ardinghelli</td>
</tr>
<tr>
<td></td>
<td>anger_jones</td>
</tr>
<tr>
<td></td>
<td>nervous_montalcini</td>
</tr>
</tbody>
</table>

Winner (28 days / 406) | Second (12 days / 78) | Third (11 days / 66)
Strawberry scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>id</th>
<th>Name</th>
<th>Strawberries Peak</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>777</td>
<td>adoring_poitras</td>
<td>406</td>
<td>cryptolux</td>
</tr>
<tr>
<td>#2</td>
<td>815</td>
<td>competent_agnesi</td>
<td>78</td>
<td>grothendieck</td>
</tr>
<tr>
<td>#3</td>
<td>753</td>
<td>bright_morse</td>
<td>66</td>
<td>sebastien-riou</td>
</tr>
<tr>
<td>#4</td>
<td>877</td>
<td>vibrant_goldberg</td>
<td>55</td>
<td>chaes</td>
</tr>
<tr>
<td>#5</td>
<td>845</td>
<td>hungry_clarke</td>
<td>36</td>
<td>team4</td>
</tr>
</tbody>
</table>
Strawberry scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>id</th>
<th>Name</th>
<th>Strawberries Peak</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>777</td>
<td>adoring_poitras</td>
<td>196</td>
<td>cryptolux</td>
</tr>
<tr>
<td>#2</td>
<td>815</td>
<td>compuastert3</td>
<td>136</td>
<td>hendieck</td>
</tr>
<tr>
<td>#3</td>
<td>753</td>
<td>bright_mcdonald</td>
<td></td>
<td>sebastien-riou</td>
</tr>
<tr>
<td>#4</td>
<td>877</td>
<td>vibrant_goldberg</td>
<td>55</td>
<td>chaes</td>
</tr>
<tr>
<td>#5</td>
<td>845</td>
<td>hungry_clarke</td>
<td>36</td>
<td>team4</td>
</tr>
</tbody>
</table>

**Winners:**
Alex Biryukov
Aleksei Udovenko (U. Luxembourg)
Strawberry scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>Id</th>
<th>Name</th>
<th>Strawberries Peak</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>777</td>
<td>adoring_poitras</td>
<td>406</td>
<td>cryptolux</td>
</tr>
<tr>
<td>#2</td>
<td>815</td>
<td>competent_agnesi</td>
<td></td>
<td>grothendieck</td>
</tr>
<tr>
<td>#3</td>
<td>753</td>
<td>bright_modish</td>
<td></td>
<td>bastien-riou</td>
</tr>
<tr>
<td>#4</td>
<td>877</td>
<td>vibrant_goldberg</td>
<td></td>
<td>chaes</td>
</tr>
<tr>
<td>#5</td>
<td>845</td>
<td>hungry_clarke</td>
<td></td>
<td>team4</td>
</tr>
</tbody>
</table>
Strawberry scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>id</th>
<th>Name</th>
<th>Strawberries Peak</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>777</td>
<td>adoring_poitras</td>
<td>406</td>
<td>cryptolux</td>
</tr>
<tr>
<td>#2</td>
<td>815</td>
<td>competent_agnesi</td>
<td>78</td>
<td>grothendieck</td>
</tr>
<tr>
<td>#3</td>
<td>753</td>
<td>bright_morse</td>
<td>66</td>
<td>sebastien-riou</td>
</tr>
<tr>
<td>#4</td>
<td>877</td>
<td></td>
<td></td>
<td>chaes</td>
</tr>
<tr>
<td>#5</td>
<td>845</td>
<td></td>
<td></td>
<td>team4</td>
</tr>
</tbody>
</table>
Banana scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>User</th>
<th>Bananas</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>team_cryptoe Experts</td>
<td>406</td>
</tr>
<tr>
<td>#2</td>
<td>cryptolux</td>
<td>78</td>
</tr>
<tr>
<td>#3</td>
<td>You!</td>
<td>55</td>
</tr>
<tr>
<td>#4</td>
<td>Team Megaloblastt</td>
<td>44</td>
</tr>
<tr>
<td>#5</td>
<td>jean_onche</td>
<td>28</td>
</tr>
</tbody>
</table>
## Banana scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>User</th>
<th>Bananas</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>team_cryptoeXperts</td>
<td>406</td>
</tr>
<tr>
<td>#2</td>
<td>cryptolux</td>
<td>386</td>
</tr>
<tr>
<td>#3</td>
<td>You!</td>
<td>386</td>
</tr>
<tr>
<td>#4</td>
<td>Team Megaloblast</td>
<td>44</td>
</tr>
<tr>
<td>#5</td>
<td>jean_onche</td>
<td>28</td>
</tr>
</tbody>
</table>

**Winners:** Louis Goubin, Pascal Paillier, Matthieu Rivain, Junwei Wang (CryptoExperts)
Banana scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>User</th>
<th>Bananas</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>team_cryptoe</td>
<td>406</td>
</tr>
<tr>
<td>#2</td>
<td>cryptolux</td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td>You!</td>
<td></td>
</tr>
<tr>
<td>#4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We didn’t cheat!

- We didn’t host the server
- Sources on GitHub
- Junwei’s presentation next
- White-paper coming soon
# Banana scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>User</th>
<th>Bananas</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>team_cryptoe xperts</td>
<td>406</td>
</tr>
<tr>
<td>#2</td>
<td>cryptolux</td>
<td>78</td>
</tr>
<tr>
<td>#3</td>
<td>You!</td>
<td>55</td>
</tr>
<tr>
<td>#4</td>
<td>Team Megaloblastt</td>
<td>44</td>
</tr>
<tr>
<td>#5</td>
<td>jean_onche</td>
<td>28</td>
</tr>
</tbody>
</table>

Alex Biryukov
Aleksei Udovenko (U. Luxembourg)
## Banana scoreboard

<table>
<thead>
<tr>
<th>Rank</th>
<th>User</th>
<th>Bananas</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>team_cryptoe...</td>
<td>406</td>
</tr>
<tr>
<td>#2</td>
<td>cryptolux</td>
<td>78</td>
</tr>
<tr>
<td>#3</td>
<td>You!</td>
<td></td>
</tr>
<tr>
<td>#4</td>
<td>Team Megalobl...</td>
<td></td>
</tr>
<tr>
<td>#5</td>
<td>jean_onche</td>
<td>28</td>
</tr>
</tbody>
</table>
Reveal Secrets in Adoring Poitras
A victory of reverse engineering and cryptanalysis over challenge 777

Louis Goubin\textsuperscript{1,4} Pascal Paillier\textsuperscript{1}
Matthieu Rivain\textsuperscript{1} Junwei Wang\textsuperscript{1,2,3}

\textsuperscript{1}CryptoExperts
\textsuperscript{2}University of Luxembourg
\textsuperscript{3}University of Paris 8
\textsuperscript{4}University of Versailles-St-Quentin-en-Yvelines

CHES 2017, Rump Session, Taipei
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Browsers stuck at loading it...
Editors are broken by it...
Some compilers (e.g., llvm) keep compiling and reporting warnings...

777 broke my editor 😞

So was my compiler.
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Untidy Code

More than 1k functions

random naming

0x3fff

not used

duplicate
Readability Processing

- Duplicate / redundancy / unused codes elimination
- Functions / variables renaming
- Constants rewriting
- Code combination

Only 20 functions are remaining
Readability Processing

- Duplicate / redundancy / unused codes elimination
- Functions / variables renaming
- Constants rewriting
- Code combination

Only 20 functions are remaining

```c
void copy(uint KLedCyW, uint RjmyJ) {int_arr[(a+KLedCyW) & 0x3fff] = int_arr[RjmyJ & 0x3fff];}
void encode(uint owhj0, uint nBqXn) {assign(owhj0, in_ptr[nBqXn]);}
void decode(uint hFqe10, uint jvXpt) {out_ptr[hFqe10]=int_arr[(a+jvXpt) & 0x3fff];}
void rshift_xor(uint HCCOL, uint ISRfIdp, uint uYFMX) {int_arr[HCCOL&0x3fff]^=16(int_arr[a+ISRfIdp] & 0x3fff) >> ufYFMX); /*print}
void lshift_xor(uint NCJbw, uint MQQXG, uint AyoLZf) {uint dyfs=(int_arr[(a+AyoLZf) & 0x3fff]) & int_arr[(a+NCJbw) & 0x3fff]^=d;)
void expand_bit(uint SteQ1D, uint ZupEP, uint Z1Qz) {int_arr[(a+SteQ1D) & 0x3fff]=((int_arr[(a+ZupEP) & 0x3fff] >> Z1Qz) & 1);}

uint lookup1(uint AKBKig) {return int_arr[(a+AKBKig) & 0x3fff];}
uint lookup2(uint WAvV, uint ZcvD) {return int_arr[(WAvV & ZcvD) & 0x3fff];}
void assign(uint Ubejj, uint UmmwjUh0) {int_arr[(a+Ubejj) & 0x3fff]=UmmwjUh0;}
void assign_a(uint wEOx) {a = wEOx;}
void assign_b(uint fnmqXl) {b=int_arr[fnmqXl] & 0x7fff;}
void update_a() {a=lookup2(1592,mix(b)); print("%lu\n",a);}
void update_b() {b=0x7fff&lookup2(522,mix(b));}

void mistery(uint wJxea, uint QBGXUN) {uint t = (~int_arr[(a+QBGXUN) & 0x3fff]) & 0x7fff; assign(wJxea,lookup2(2979,mix(t)));}
// bitwise operation
void xor(uint oEHwkw, uint KCZu, uint MtCA) {int_arr[(a+oEHwkw) & 0x3fff]=int_arr[(a+KCZu) & 0x3fff] ^ int_arr[(a+MtCA) & 0x3fff];}
void and(uint bmnp, uint UNFg, uint PqCtZY) {uint t = int_arr[(a+UNFg) & 0x3fff] & int_arr[(a+PqCtZY) & 0x3fff] & 0x3fff;)
void or(uint eTGl, uint udoXFs, uint mezPNN) {int_arr[(a+eTGl) & 0x3fff] | int_arr[(a+udoXFs) & 0x3fff] | int_arr[(a+mezPNN) & 0x3fff];}
void not(uint YfnT, uint JKTW) {int_arr[(a+YfnT) & 0x3fff] = ~int_arr[(a+JKTW) & 0x3fff];}
// jump
void goto_f(uint LKh0C) {pc = bop + LKh0C;}
void jump_if(uint DbvJ0, uint FleFNlf, uint LeHf) { if((lookup2(2979,mix(b)) == lookup2(DbvJ0, FleFNlf)) || count >= 64) {printf("%d");
```
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Universal Turing Machine

```
void call(void)
{
    unYAQ = klspCVy;
    sutlnu = klspCVy + sizeof(JGNNvi)/sizeof(uchar);
    JqcadL = klspCVy;
    while (JqcadL < sutlnu) {
        uchar eMrr = *JqcadL++;
        if (eMrr == 0) {
            void (*QIEb)();
            QIEb = (void*)funcptrs[*JqcadL++];
            uint *AnezsV = (uint*)JqcadL;
            JqcadL += eMrr*8;
            QIEb();
        } else if (eMrr == 1) {
            void (*QIEb)();
            QIEb = (void*)funcptrs[*JqcadL++];
            uint *AnezsV = (uint*)JqcadL;
            JqcadL += eMrr*8;
            QIEb(AnezsV[0]);
        }
    }
}
```
Universal Turing Machine (2)

\[ \Rightarrow \text{UTM(RASP)} \]
Universal Turing Machine (3)
De-virtualization - Simulate the UTM

```c
else if (eMmr == 3) {
    void (*QiEb)(uint, uint, uint);
    QiEb = (void*)funcptrs[*pc++];
    uint *AnezsV = (uint*)pc;
    pc += eMmr*8;
    // QiEb(AnezsV[0], AnezsV[1], AnezsV[2]);
    #ifdef SIMULATE
        printf("%8s(%d,%d,%d);\n",flist[*(pc-1-eMmr*8)],AnezsV[0],AnezsV[1],AnezsV[2]);
    #endif
}
else if (eMmr == 4) {
    void (*QiEb)(uint, uint, uint, uint);
    QiEb = (void*)funcptrs[*pc++];
    uint *AnezsV = (uint*)pc;
    pc += eMmr*8;
    // QiEb(AnezsV[0], AnezsV[1], AnezsV[2], AnezsV[3]);
    #ifdef SIMULATE
        printf("%8s(%d,%d,%d,%d);\n",flist[*(pc-1-eMmr*8)],AnezsV[0],AnezsV[1],AnezsV[2],AnezsV[3]);
    #endif
}
```
De-virtualization - Simulate the UTM

We get a bitwise-based program (600k operations).
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Bitwise-based program

Input: plaintext bits \((b_1, b_2, \cdots, b_{128})\)
Output: ciphertext bits \((c_1, c_2, \cdots, c_{128})\)

for \(i = 1\) to \(128\) do
  \(t[addr_1,i] \leftarrow 0b\ b_i\ b_i\ b_i\ \cdots\ b_i\)  \(\triangleright\) expand \(b_i\) to unsigned long integer (64 bits)
  for \(j = 1\) to \(64\) do
    \(t[addr_2,i + j \times 2^{12}] \leftarrow t[addr_1,i]\)
  end for
end for

\textbf{BitwiseOperationLoop1}
\textbf{BitwiseOperationLoop2}
\ldots
\textbf{BitwiseOperationLoop2573}

for \(i = 1\) to \(129\) do
  \(t[addr_3,i] \leftarrow v_i\)  \(\triangleright\) \(v_i \in GF(2)\) is a constant
  for \(j = 1\) to \(64\) do
    \(tmp \leftarrow t[addr_4,i + j \times 2^{12}] \oplus t[addr_5,i + j \times 2^{12}]\)
    \(t[addr_3,i] \leftarrow t[addr_3,i] \oplus \text{Parity}(tmp)\)  \(\triangleright\) \text{Parity} computes the number of 1-bit modulo 2
  end for
end for

\textbf{BitwiseOperationLoop2574}
\ldots
\textbf{BitwiseOperationLoop2582}

for \(i = 1\) to \(128\) do
  \(c_i \leftarrow t[addr_6,i]\)
end for
From Bitwise Program to Boolean Circuits

- 64 (loop length) * 64 (number of bits in a unsigned long integer) independent AES computations operated in boolean circuits
- 3 out of 64\times64 are the real and identical AES computations (e.g., bit 42 of loop 26)
- Hence, the bitwise-based program can be simplified as a boolean circuits with 600k gates (XOR, AND, OR, NOT).
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Boolean Circuits Minimization

- Constant variable detection and propagation
- Dead code elimination
- Deduplication
- “Potential” pseudorandomness detection and removal
- Repeat the above steps until no more constant / duplicate / "potential" pseudorandomness can be detected

Finally, the circuits is reduced to 280k boolean gates (53% smaller)
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Data Dependency Graph (DDG)

\[
x = a; \\
y = b; \\
x = y + x; \\
y = x \times y; \\
z = x - y; \\
x = z \times x;
\]
DDG of the Circuits (First 5%)
First Round Computation of AES

MixColumns
SubBytes
Extracting the Branches (Clustering)
Outline

0. Downloading and Compiling the Code
1. Cleaning the Code
2. De-Virtualization
3. From Bitwise Program to Boolean Circuits
4. Boolean Circuits Minimization
5. Data Dependency Analysis
6. Algebraic Analysis
Assumption

Assumption (Informal)
Each of the green "branch" corresponds to an individual S-Box computation in the first round of AES, the $t$-bit output $(s_1, s_2, \cdots, s_t)$ of which is a linear encoding of a real S-Box output bit.
Output Bits of A Branch

Bits in a branch (530)
S-Box output bits (34)
Solve a Systems of Linear Equations

\[
\begin{bmatrix}
  s_1^{(1)} & s_2^{(1)} & \ldots & s_{34}^{(1)} & 1 \\
  s_1^{(2)} & s_2^{(2)} & \ldots & s_{34}^{(2)} & 1 \\
  \vdots & \vdots & \ddots & \vdots & \vdots \\
  s_1^{(n)} & s_2^{(n)} & \ldots & s_{34}^{(n)} & 1 \\
\end{bmatrix}
\begin{bmatrix}
  a_1 \\
  a_2 \\
  \vdots \\
  a_{34} \\
  a_{35} \\
\end{bmatrix}
= 
\begin{bmatrix}
  \text{SBox}(x^{(1)} \oplus \hat{k})[i] \\
  \text{SBox}(x^{(2)} \oplus \hat{k})[i] \\
  \vdots \\
  \text{SBox}(x^{(n)} \oplus \hat{k})[i] \\
\end{bmatrix}
\]

If \( n \geq 35 + 8 + \lambda \), \( \Pr["\hat{k} \neq k^* \text{ has a solution"}] \leq 2^{-\lambda} \).
Results

```
In[488]: LinearBreak[data]

key=0x0
key=0x10
key=0x20
key=0x30
key=0x40
key=0x50
key=0x60
key=0x70
key=0x80
key=0x90
key=0xa0
key=0xb0
key=0xc0

!!!!!!!!!!!!!! 2 - 0 - 0xcf !!!!!!!!!!!!!!!
!!!!!!!!!!!!!! 2 - 1 - 0xcf !!!!!!!!!!!!!!!
!!!!!!!!!!!!!! 2 - 2 - 0xcf !!!!!!!!!!!!!!!
!!!!!!!!!!!!!! 2 - 3 - 0xcf !!!!!!!!!!!!!!!
!!!!!!!!!!!!!! 2 - 4 - 0xcf !!!!!!!!!!!!!!!
!!!!!!!!!!!!!! 2 - 5 - 0xcf !!!!!!!!!!!!!!!
!!!!!!!!!!!!!! 2 - 6 - 0xcf !!!!!!!!!!!!!!!
!!!!!!!!!!!!!! 2 - 7 - 0xcf !!!!!!!!!!!!!!!
key=0xd0
key=0xe0
key=0xf0
```
Thank you!