The Key Recovery Attacks against Commercial White Box Cryptography implementations

Sanghwan Ahn(h2spice), LINE Corporation

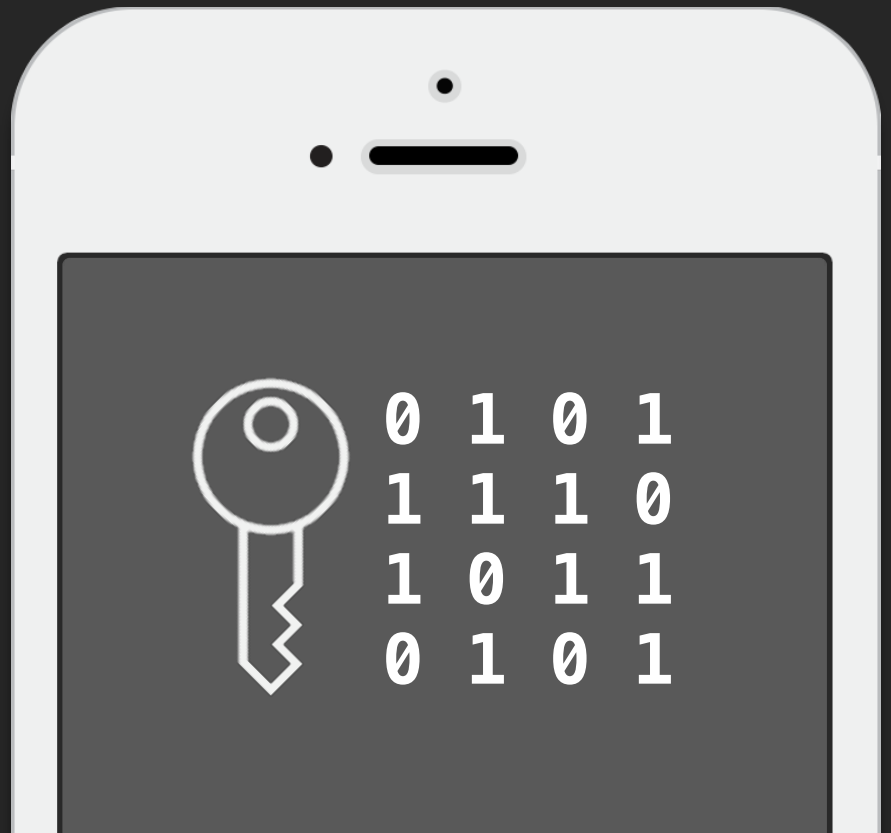
LINE

Agenda

- Introduction
- White-box Cryptography
- White-box Cryptanalysis
- Conclusion

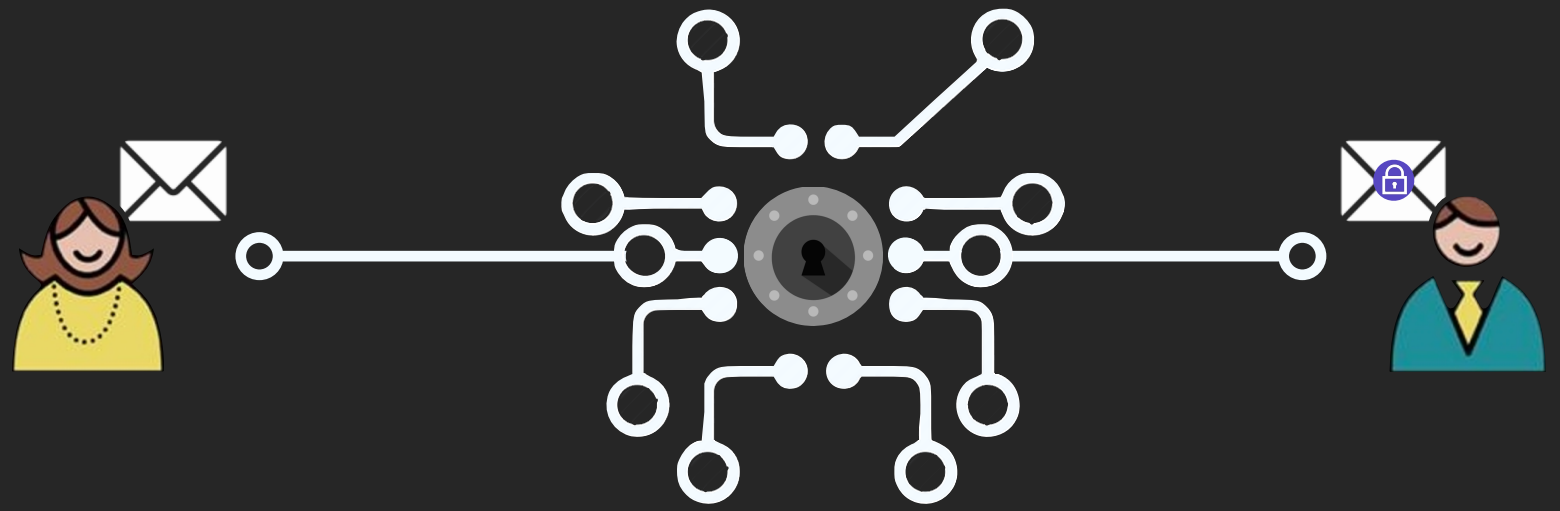
INTRODUCTION

Crypto system In the Software



WHITE-BOX THREAT MODEL

- Binary is completely visible to an attacker
- Attacker has full access to the cryptography algorithm
- Attacker has full control over its execution environment
- Unlimited amount of queries



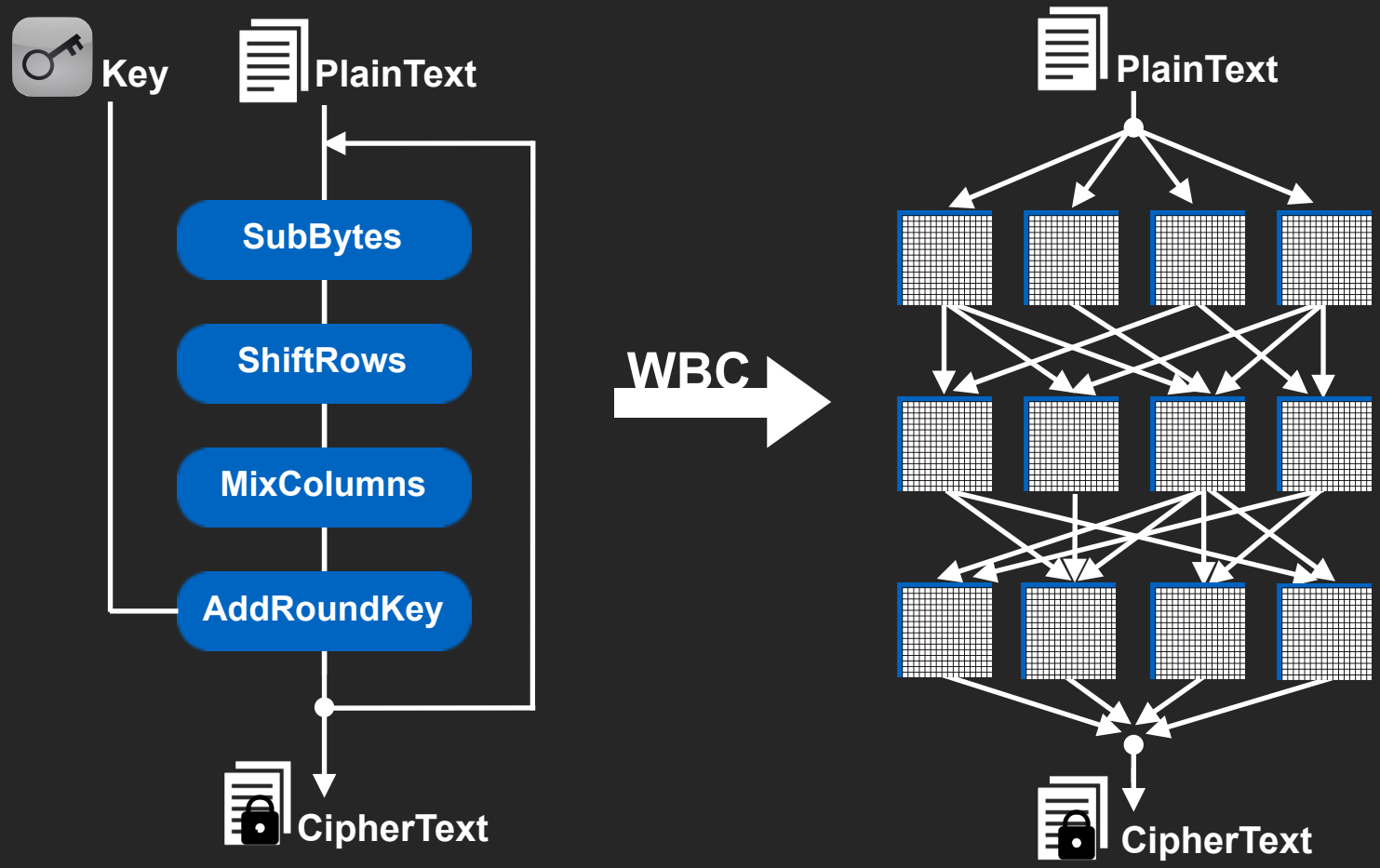
SECURE ELEMENT

- Trusted execution environment(TEE)
  - ARM Trustzone, Intel SGX, AMD Memory Encryption
  - It’s almost safe, but not many supported devices (mostly latest devices)

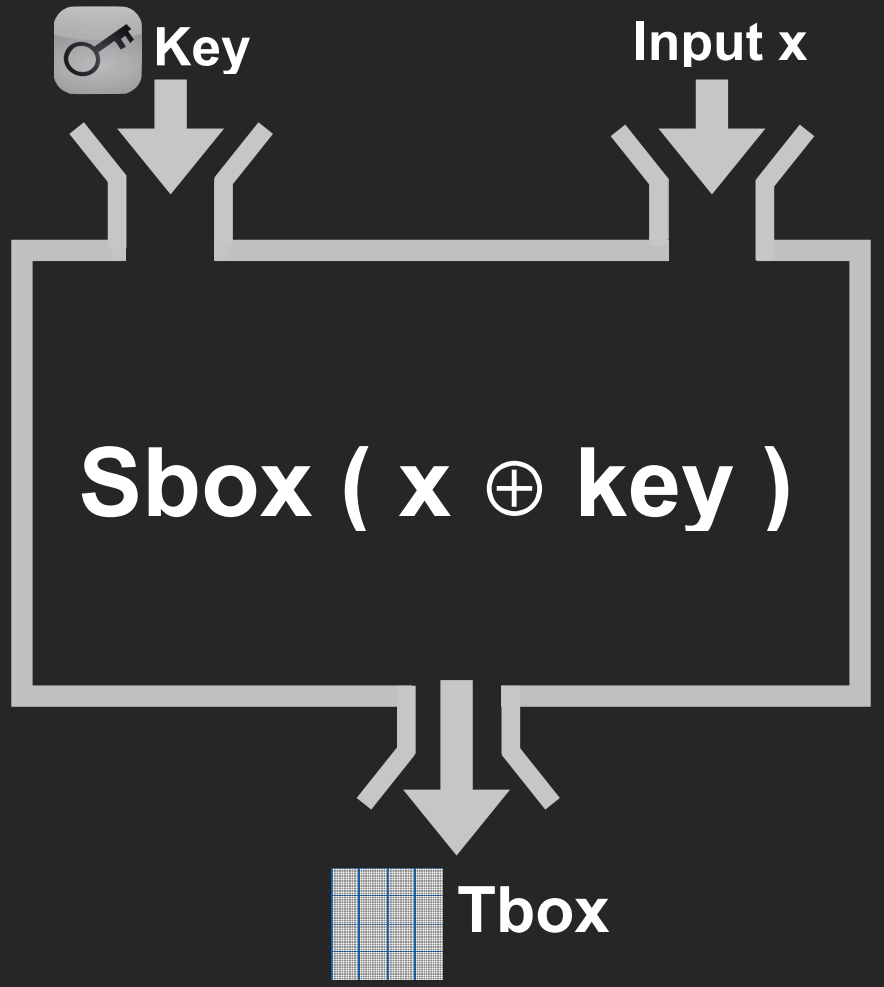
- White-box cryptography(WBC)
  - All academic WBC solutions have been broken
  - No attack has been observed to date on commercial WBC

White-box Cryptography

White-box AES Implementation

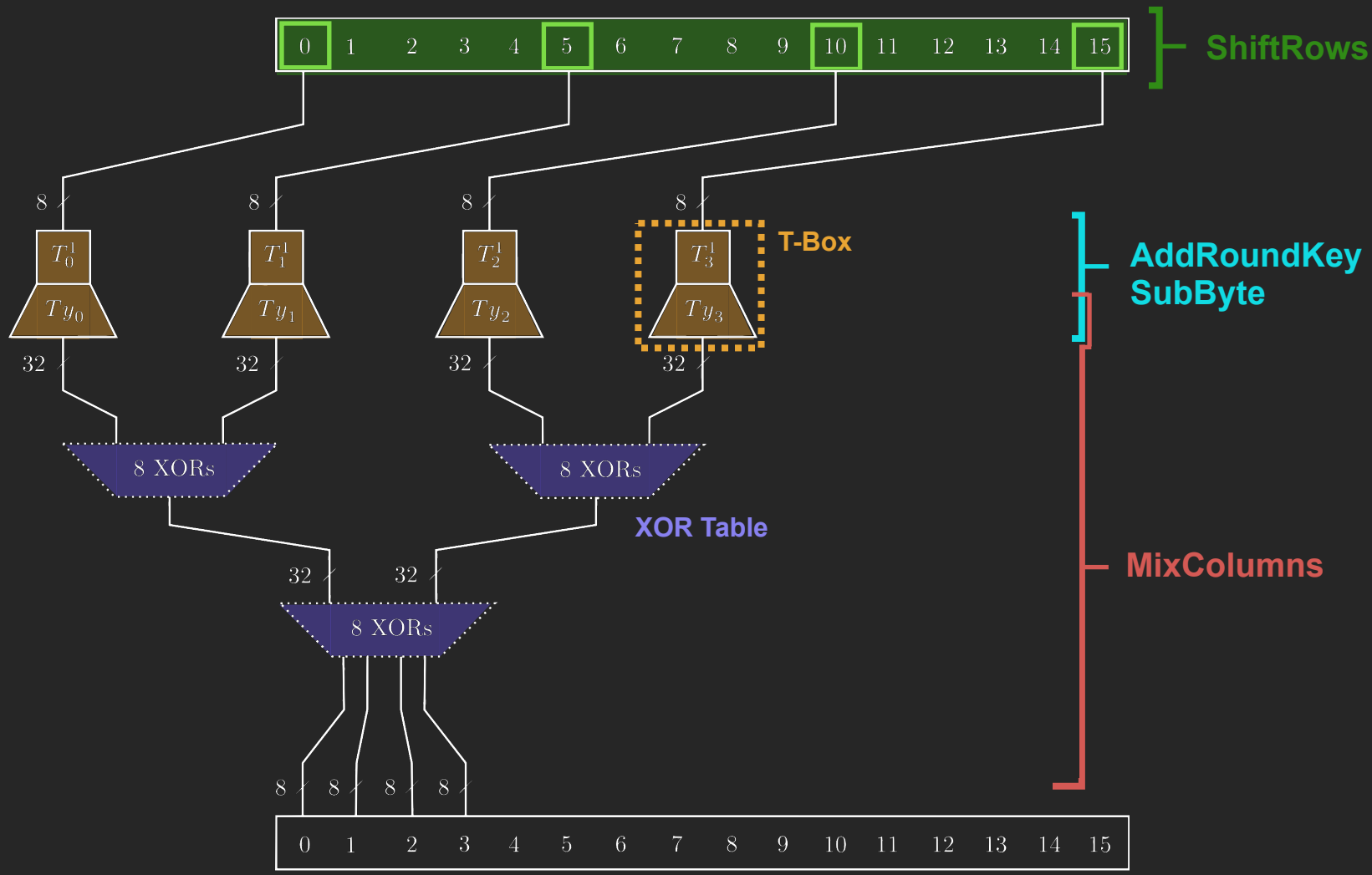


KEY PARTIAL EVALUATION



```
unsigned int Tboxes[9][16][256] =
{
    /*First Round, K[0] is 0x6c*/
    {
        0xf05050a0, 0x443c3c78, 0xba9f9f25, 0xe3a8a84b, 0xcf45458a, 0x10f9f9e9, 0x06020204, 0x817f7ffe,
        0xc5434386, 0xd74d4d9a, 0x55333366, 0x94858511, 0x6bd0d0bb, 0x2aefefc5, 0xe5aaaa4f, 0x16fbfbed,
        0x30101020, 0x1affffe5, 0x0ef3f3fd, 0x6dd2d2bf, 0xdfbcbc63, 0xc1b6b677, 0x75dadaaf, 0x63212142,
        0xad92923f, 0xbc9d9d21, 0x48383870, 0x04f5f5f1, 0xf35151a2, 0xfea3a35d, 0xc0404080, 0x8a8f8f05,
        0x7b292952, 0x3ee3e3dd, 0x712f2f5e, 0x97848413, 0xf65252a4, 0x4d3b3b76, 0x61d6d6b7, 0xceb3b37d,
        0x2d1b1b36, 0xb26e6edc, 0xee5a5ab4, 0xfba0a05b, 0x1b090912, 0x9e83831d, 0x742c2c58, 0x2e1a1a34,
        0xde4a4a94, 0xd44c4c98, 0xe85858b0, 0x4acfcf85, 0xbe6a6ad4, 0x46cbcb8d, 0xd9bebe67, 0x4b393972,
        0x60202040, 0x1ffcfce3, 0xc8b1b179, 0xed5b5bb6, 0xf55353a6, 0x68d1d1b9, 0x00000000, 0x2cededc1,
        0x937171e2, 0x73d8d8ab, 0x53313162, 0x3f15152a, 0x5c343468, 0xf4a5a551, 0x34e5e5d1, 0x08f1f1f9,
        0x5a36366c, 0x413f3f7e, 0x02f7f7f5, 0x4fcccc83, 0xc2b7b775, 0x1cfdfde1, 0xae93933d, 0x6a26264c,
        0x26ebebcd, 0x6927274e, 0xcdb2b27f, 0x9f7575ea, 0x0907070e, 0x36121224, 0x9b80801b, 0x3de2e2df,
        0x28181830, 0xa1969637, 0x0f05050a, 0xb59a9a2f, 0x0c040408, 0x52c7c795, 0x65232346, 0x5ec3c39d,
        0x19fefee7, 0x62d7d7b5, 0xe6abab4d, 0x9a7676ec, 0x50303060, 0x03010102, 0xa96767ce, 0x7d2b2b56,
        0x0df2f2ff, 0xbd6b6bd6, 0xb16f6fde, 0x54c5c591, 0xa56363c6, 0x847c7cf8, 0x997777ee, 0x8d7b7bf6,
        0xbf9c9c23, 0xf7a4a453, 0x967272e4, 0x5bc0c09b, 0xecadad41, 0x67d4d4b3, 0xfda2a25f, 0xeaafaf45,
        0x15fafaef, 0xeb5959b2, 0xc947478e, 0x0bf0f0fb, 0x45caca8f, 0x9d82821f, 0x40c9c989, 0x877d7dfa,
        0x49cece87, 0xff5555aa, 0x78282850, 0x7adfdfa5, 0xb69b9b2d, 0x221e1e3c, 0x92878715, 0x20e9e9c9,
        0xbb6969d2, 0x70d9d9a9, 0x898e8e07, 0xa7949433, 0x38e1e1d9, 0x13f8f8eb, 0xb398982b, 0x33111122,
        0xcbb0b07b, 0xfc5454a8, 0xd6bbbb6d, 0x3a16162c, 0xc3414182, 0xb0999929, 0x772d2d5a, 0x110f0f1e,
        0xdabfbf65, 0x31e6e6d7, 0xc6424284, 0xb86868d0, 0x8f8c8c03, 0xf8a1a159, 0x80898909, 0x170d0d1a,
        0xdd4b4b96, 0xdcbdbd61, 0x868b8b0d, 0x858a8a0f, 0x23e8e8cb, 0x7cdddda1, 0x9c7474e8, 0x211f1f3e,
        0x241c1c38, 0xf1a6a657, 0xc7b4b473, 0x51c6c697, 0xd5baba6f, 0x887878f0, 0x6f25254a, 0x722e2e5c,
        0x91868617, 0x58c1c199, 0x271d1d3a, 0xb99e9e27, 0xa36161c2, 0x5f35356a, 0xf95757ae, 0xd0b9b969,
        0xd8484890, 0x05030306, 0x01f6f6f7, 0x120e0e1c, 0x907070e0, 0x423e3e7c, 0xc4b5b571, 0xaa6666cc,
        0xa8919139, 0xa4959531, 0x37e4e4d3, 0x8b7979f2, 0x5dc2c29f, 0x6ed3d3bd, 0xefacac43, 0xa66262c4,
        0xdb494992, 0x0a06060c, 0x6c242448, 0xe45c5cb8, 0x3be0e0db, 0x56323264, 0x4e3a3a74, 0x1e0a0a14,
        0xaf6565ca, 0x8e7a7af4, 0xe9aeae47, 0x18080810, 0xb46c6cd8, 0xfa5656ac, 0x07f4f4f3, 0x25eaeacf,
        0x8c8d8d01, 0x64d5d5b1, 0xd24e4e9c, 0xe0a9a949, 0x32e7e7d5, 0x43c8c88b, 0x5937376e, 0xb76d6dda,
        0xac6464c8, 0xe75d5dba, 0x2b191932, 0x957373e6, 0x57c4c493, 0xf2a7a755, 0x827e7efc, 0x473d3d7a,
        0xe15f5fbe, 0xa2979735, 0xcc444488, 0x3917172e, 0x4ccdcd81, 0x140c0c18, 0x35131326, 0x2fececc3,
        0x79dedea7, 0xe25e5ebc, 0x1d0b0b16, 0x76dbdbad, 0xca46468c, 0x29eeeec7, 0xd3b8b86b, 0x3c141428,
        0x66222244, 0x7e2a2a54, 0xab90903b, 0x8388880b, 0xa06060c0, 0x98818119, 0xd14f4f9e, 0x7fdcdca3
    },
    ...
```

Table based AES Implementation



EXAMPLE

```
void aes128_enc_wb_final(unsigned char in[16], unsigned char out[16])
{
    memcpy(out, in, 16);

    /// Let's start the encryption process now
    for (size_t i = 0; i < 9; ++i)
    {
        ShiftRows(out);

        for (size_t j = 0; j < 4; ++j)
        {
            unsigned int aa = Tyboxes[i][j * 4 + 0][out[j * 4 + 0]];
            unsigned int bb = Tyboxes[i][j * 4 + 1][out[j * 4 + 1]];
            unsigned int cc = Tyboxes[i][j * 4 + 2][out[j * 4 + 2]];
            unsigned int dd = Tyboxes[i][j * 4 + 3][out[j * 4 + 3]];

            out[j * 4 + 0] = (Txor[Txor[(aa >>  0) & 0xf][(bb >>  0) & 0xf]][Txor[(cc >>  0) & 0xf][(dd >>  0) & 0xf]]) |
                             ((Txor[Txor[(aa >>  4) & 0xf][(bb >>  4) & 0xf]][Txor[(cc >>  4) & 0xf][(dd >>  4) & 0xf]]) << 4);
            out[j * 4 + 1] = (Txor[Txor[(aa >>  8) & 0xf][(bb >>  8) & 0xf]][Txor[(cc >>  8) & 0xf][(dd >>  8) & 0xf]]) |
                             ((Txor[Txor[(aa >> 12) & 0xf][(bb >> 12) & 0xf]][Txor[(cc >> 12) & 0xf][(dd >> 12) & 0xf]]) << 4);
            out[j * 4 + 2] = (Txor[Txor[(aa >> 16) & 0xf][(bb >> 16) & 0xf]][Txor[(cc >> 16) & 0xf][(dd >> 16) & 0xf]]) |
                             ((Txor[Txor[(aa >> 20) & 0xf][(bb >> 20) & 0xf]][Txor[(cc >> 20) & 0xf][(dd >> 20) & 0xf]]) << 4);
            out[j * 4 + 3] = (Txor[Txor[(aa >> 24) & 0xf][(bb >> 24) & 0xf]][Txor[(cc >> 24) & 0xf][(dd >> 24) & 0xf]]) |
                             ((Txor[Txor[(aa >> 28) & 0xf][(bb >> 28) & 0xf]][Txor[(cc >> 28) & 0xf][(dd >> 28) & 0xf]]) << 4);
        }
    }

    /// Last round which is a bit different
    ShiftRows(out);

    for (size_t j = 0; j < 16; ++j)
    {
        unsigned char x = Tboxes_[j][out[j]];
        out[j] = x;
    }
}
```

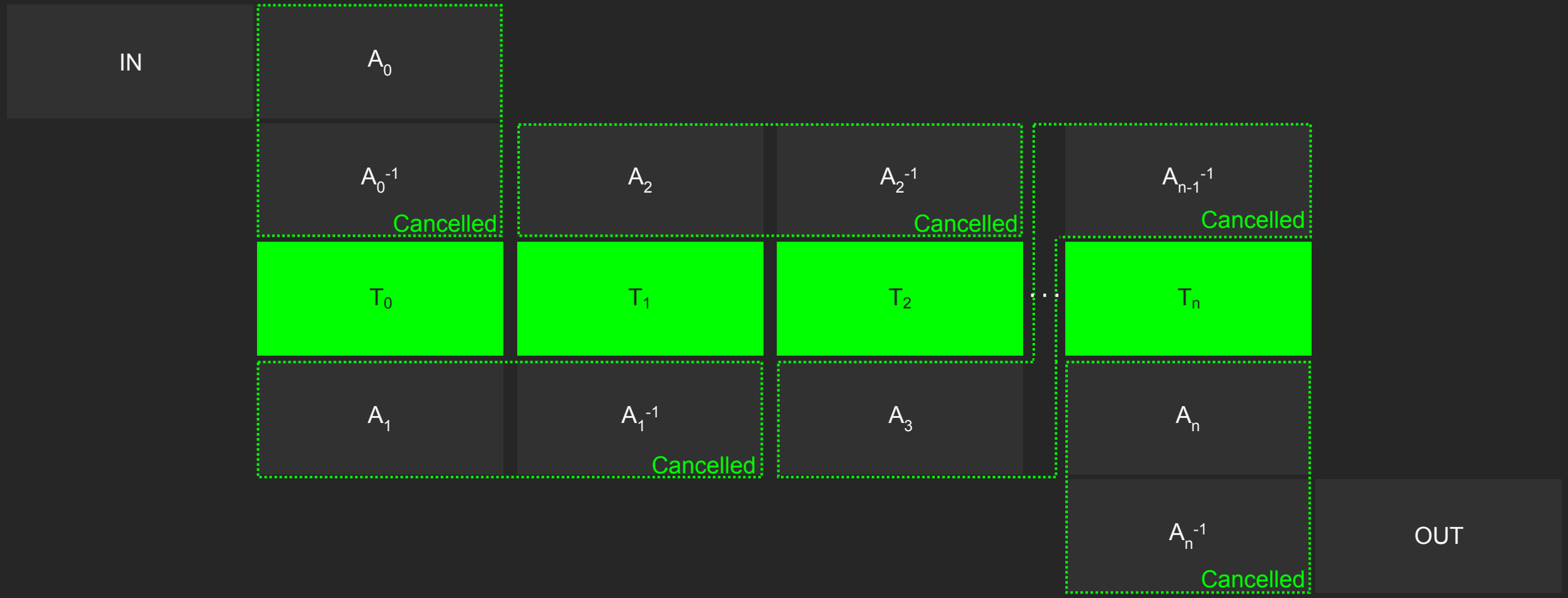
Reference : https://github.com/0vercl0k/stuffz/tree/master/wbaes_attack/wbaes128

ENCODING

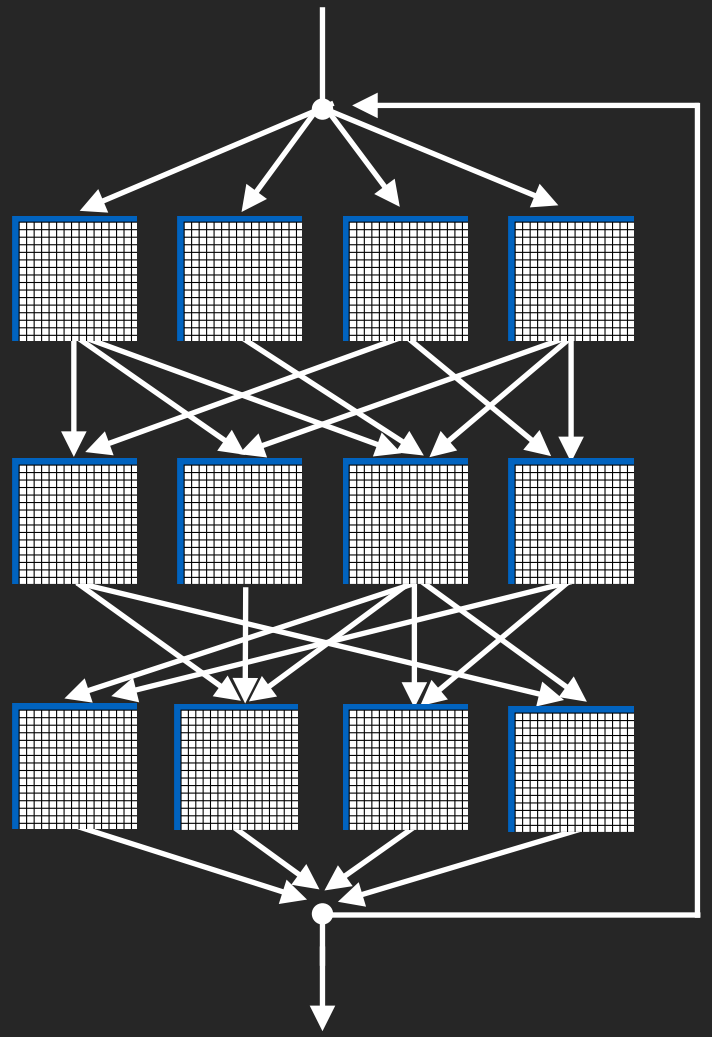
$A * A^{-1} = I_{(identity\ matrix)}$

ENCODING

$A * A^{-1} = I_{(identity\ matrix)}$

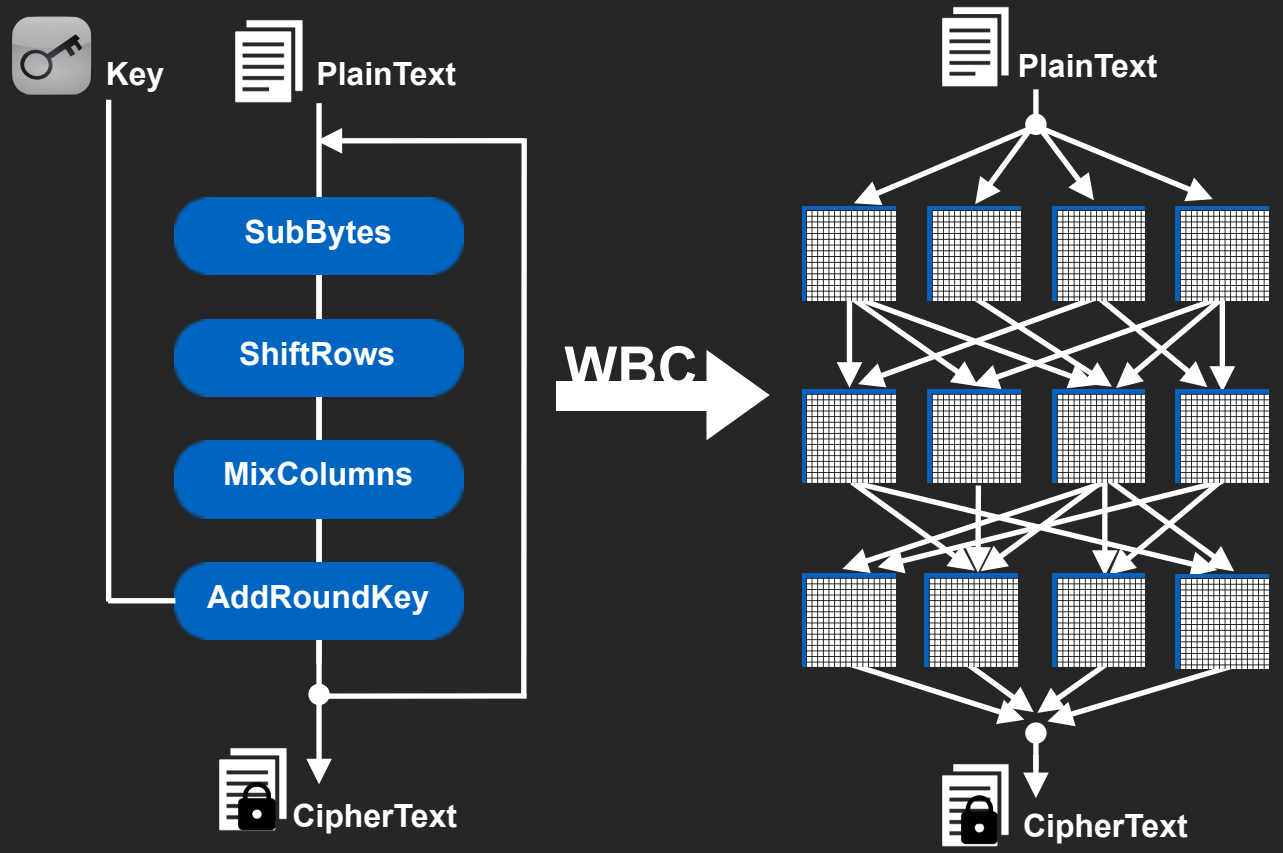


Network of look-up tables



Comparison of AES and WB-AES

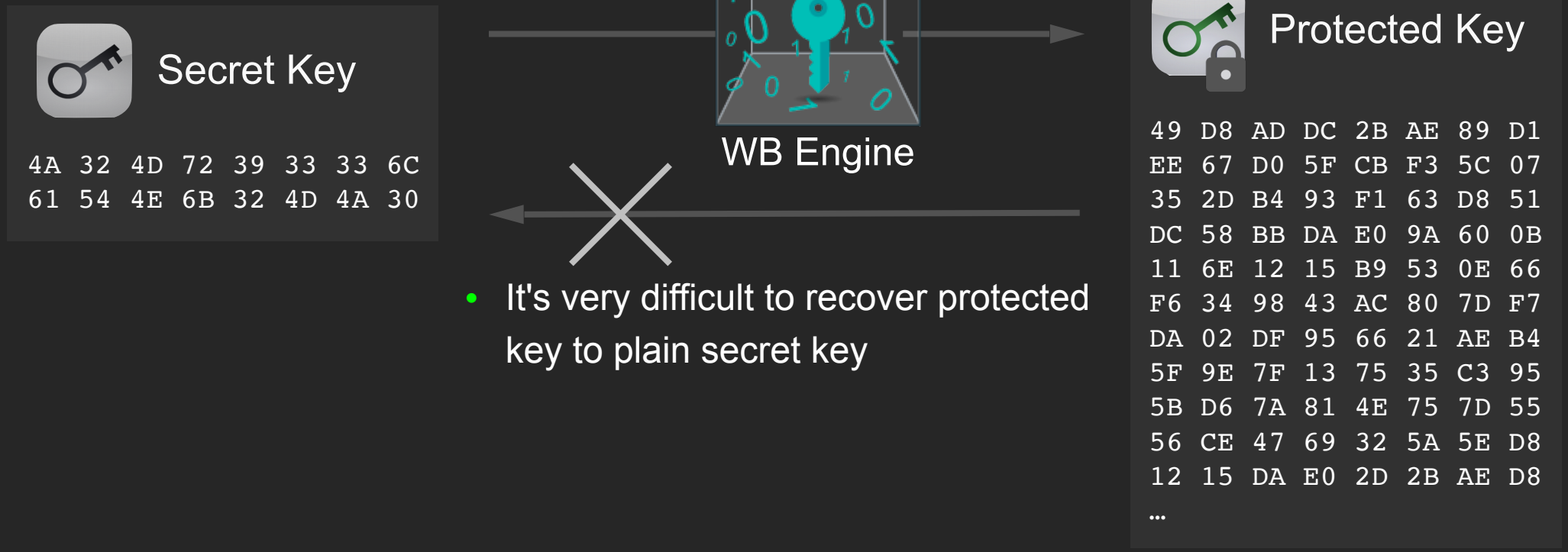
- Cryptographic key is not visible
- Cryptographic operations are obfuscated, randomized



White-box Cryptanalysis

- Academic WBC Implementations
  - Karroumi's white-box AES implementation
  - OpenWhiteBox
  - Challenges in CTF
- Commercial WBC Implementations
  - Simple cipher for performance
  - Complex cipher for security

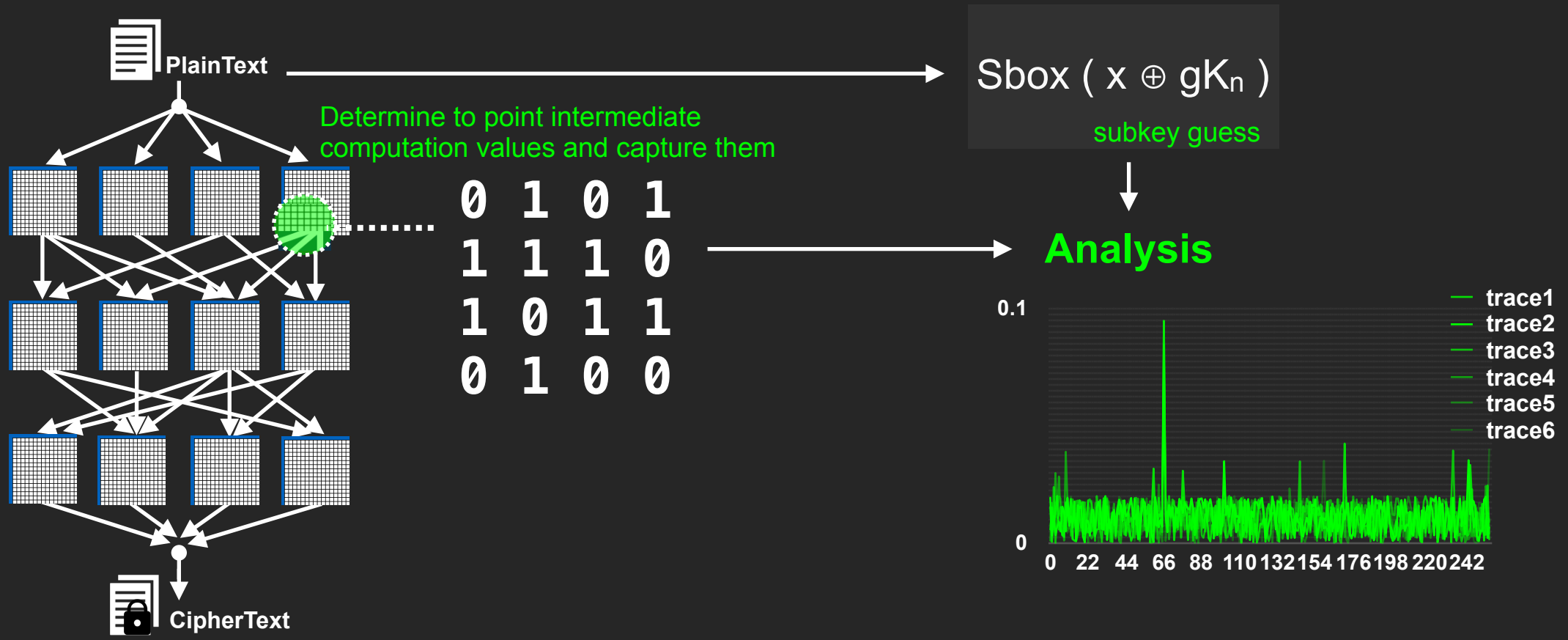
COMMERCIAL WHITE-BOX PRODUCTS



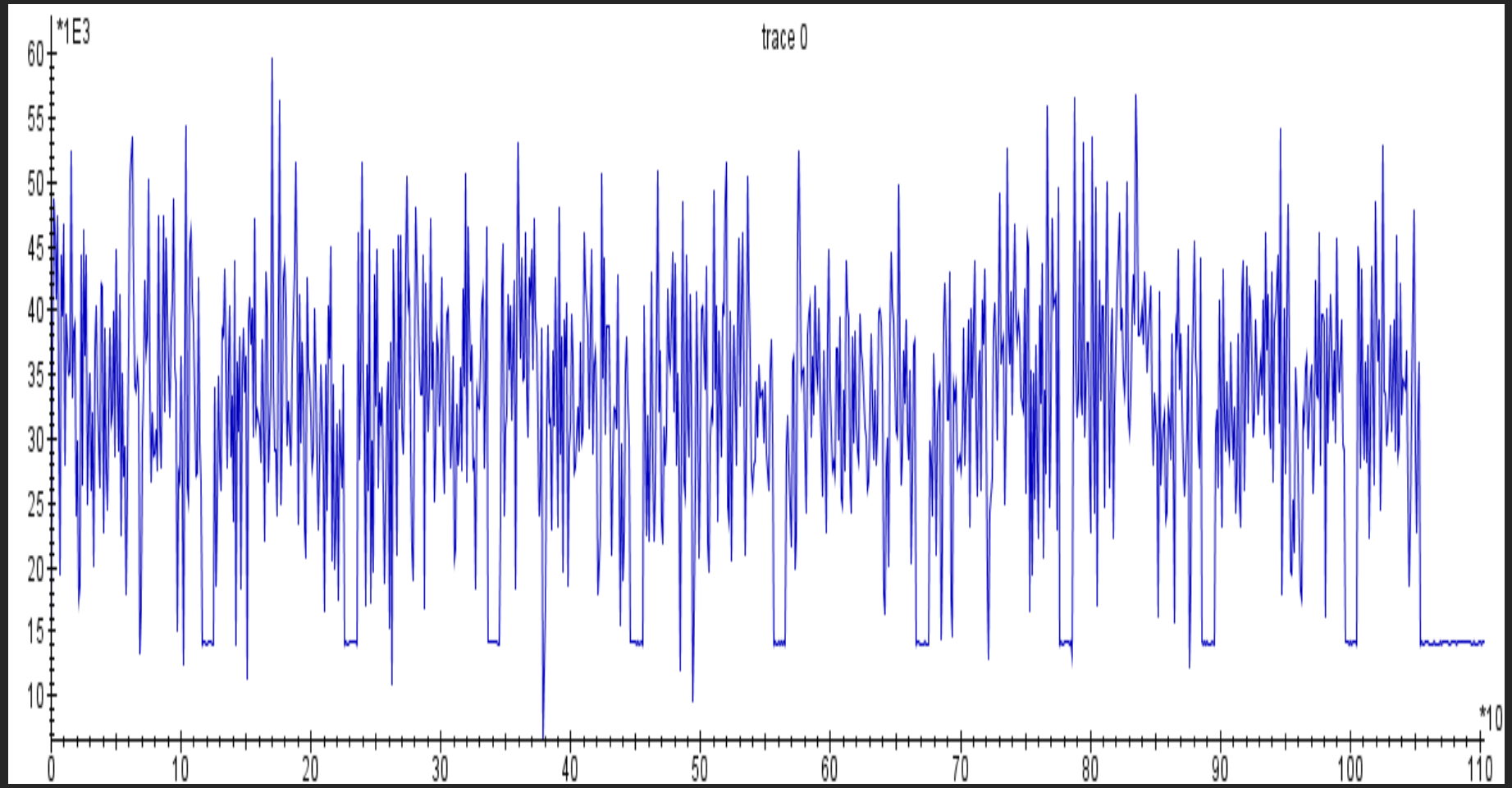
- It's very difficult to recover protected key to plain secret key

POWER ANALYSIS

- Recode intermediate computation result
- And then compare it and subkey guessed



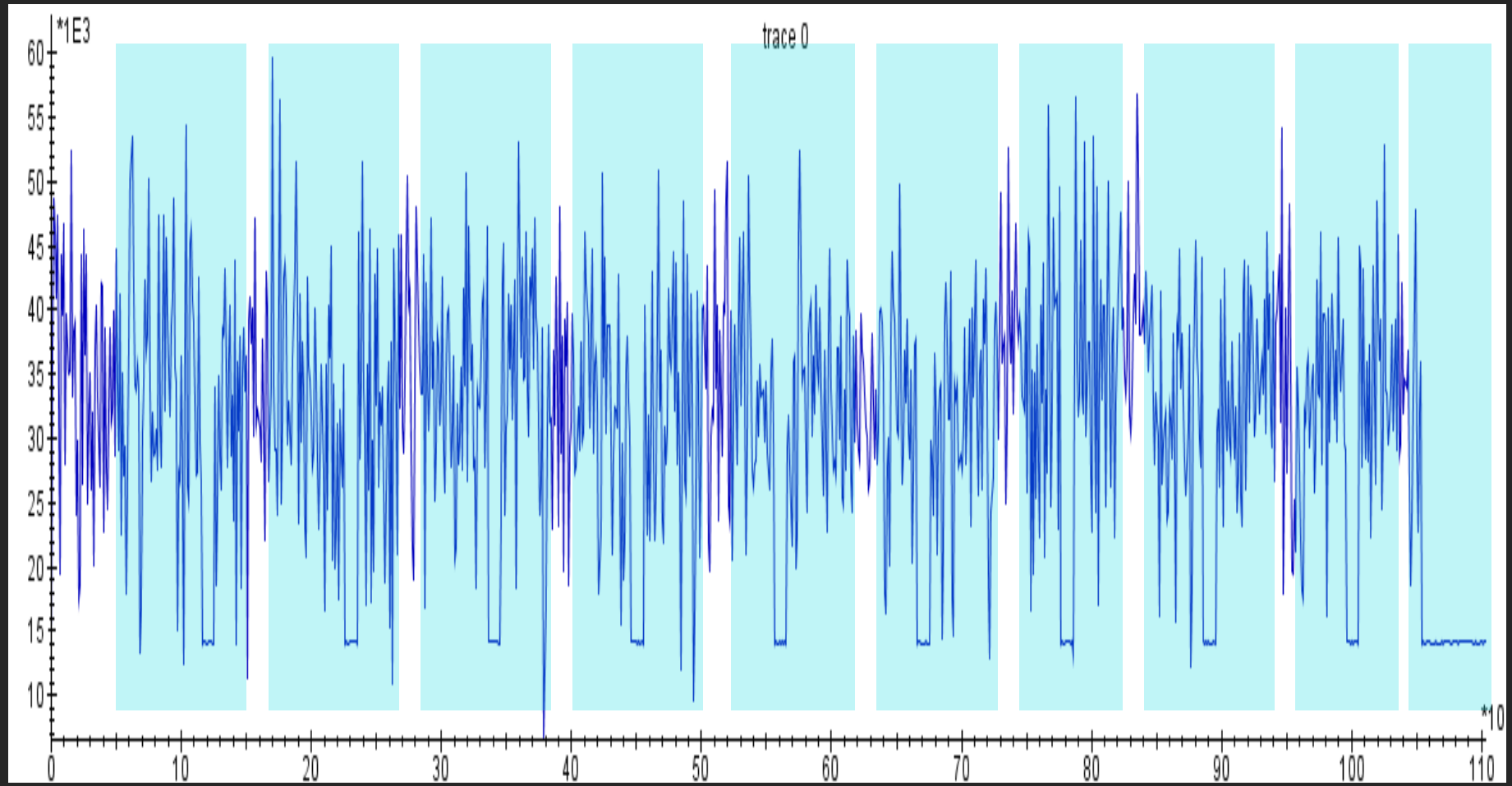
Power analysis on the hardware



Typical example of a (hardware) power trace of an unprotected AES-128 implementation (one can observe the ten rounds)

Reference : Joppe W. Bos et al. - Differential Computation Analysis: Hiding your White-Box Designs is Not Enough

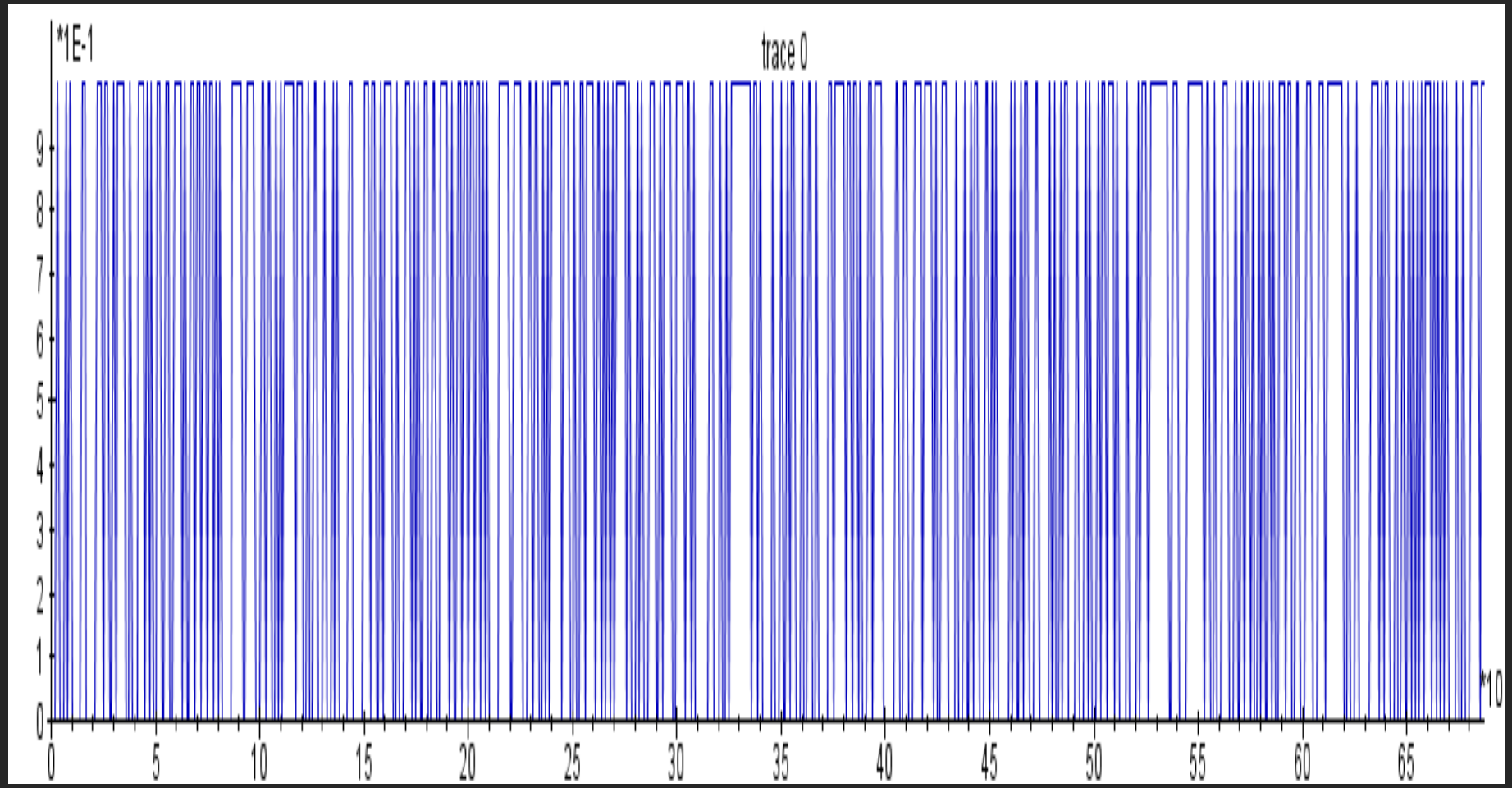
Power analysis on the hardware



Typical example of a (hardware) power trace of an unprotected AES-128 implementation (one can observe the ten rounds)

Reference : Joppe W. Bos et al. - Differential Computation Analysis: Hiding your White-Box Designs is Not Enough

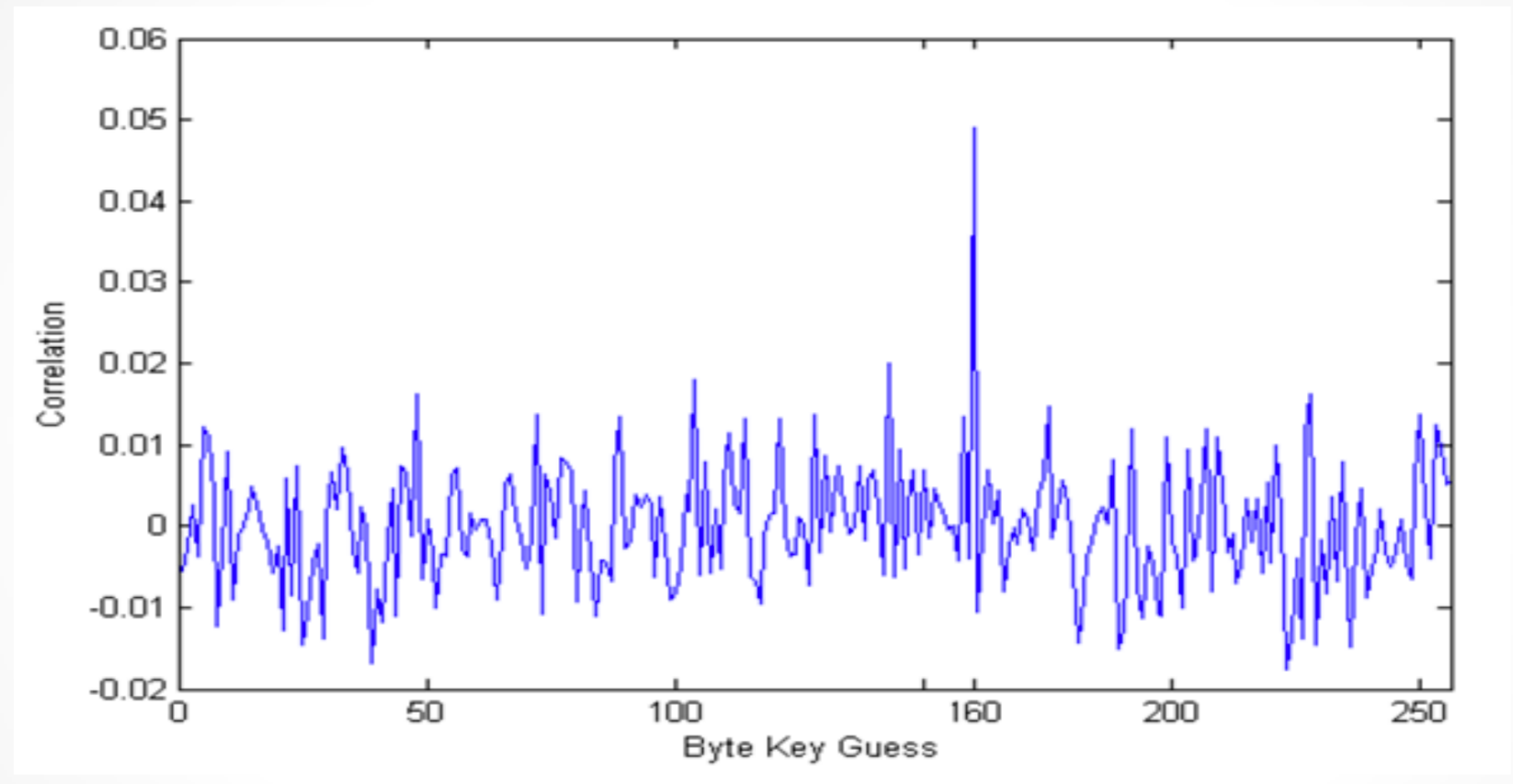
Power analysis on the software



Typical example of a portion of a serialized software trace of stack writes in an WBAES-128, with only two possible values: 0 or 1

Reference : Joppe W. Bos et al. - Differential Computation Analysis: Hiding your White-Box Designs is Not Enough

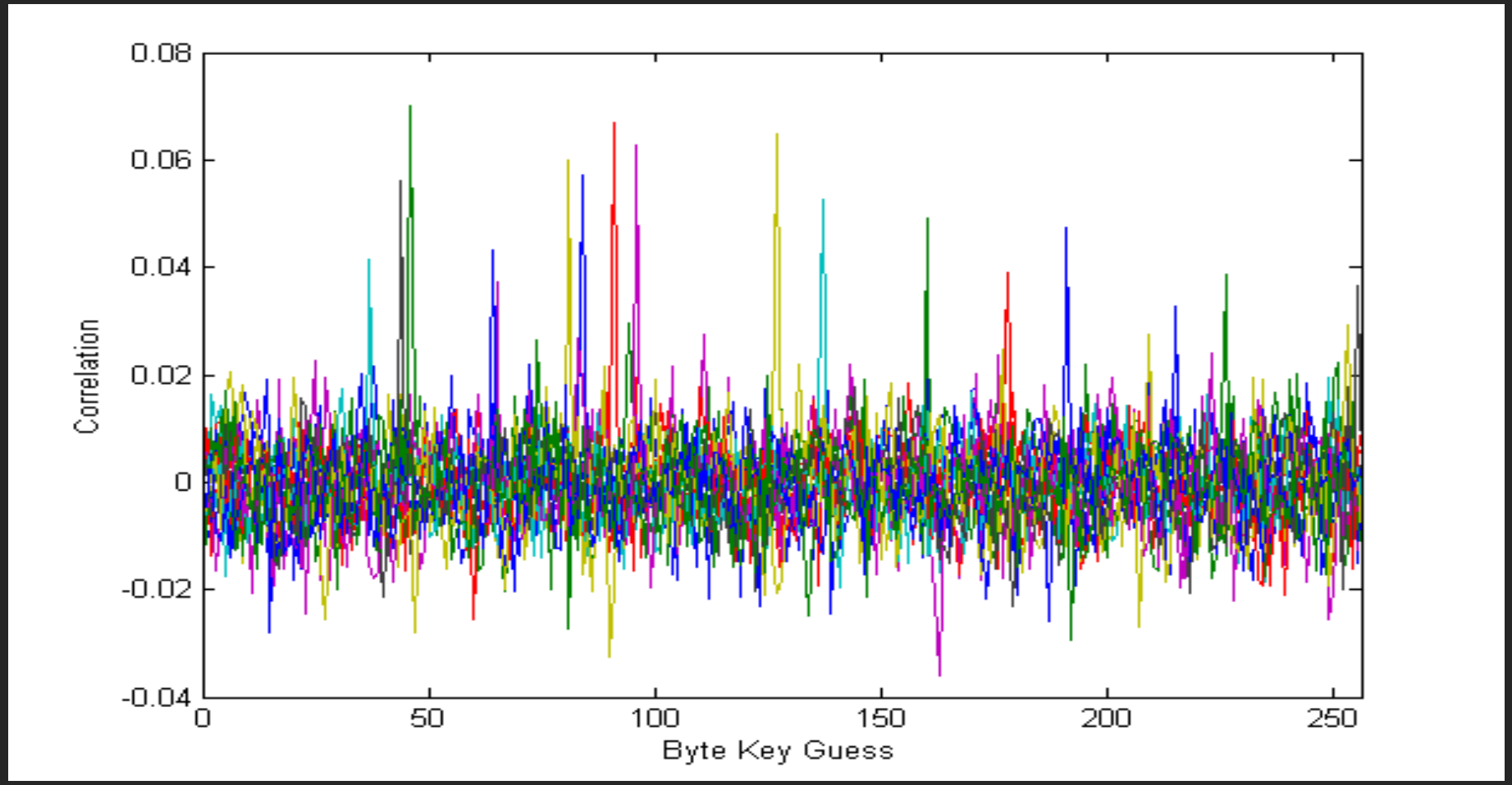
CORRELATION POWER ANALYSIS(CPA)



The correlation between the sensitive data and the power consumption for the 256 key guesses for a single byte

Reference : W. Hnath, J. Pettengill, “Differential Power Analysis Side-Channel Attacks in Cryptography,” Major Qualifying Project, Worcester Polytechnic Institute, April 2010

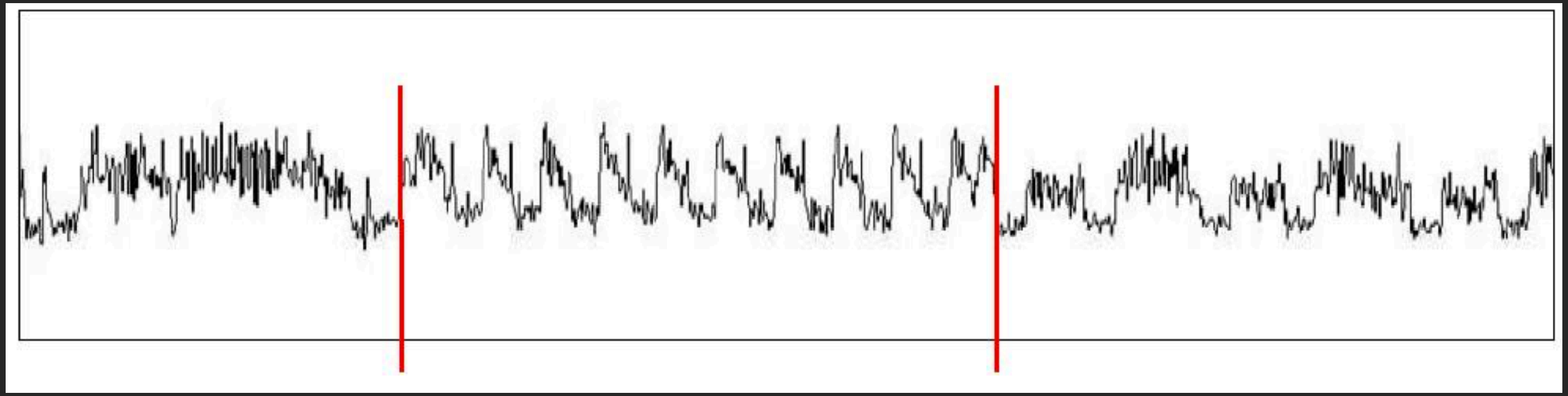
CORRELATION POWER ANALYSIS(CPA)



The correlation between the sensitive data and the power consumption for the 256 key guesses for 16 byte

Reference : W. Hnath, J. Pettengill, “Differential Power Analysis Side-Channel Attacks in Cryptography,” Major Qualifying Project, Worcester Polytechnic Institute, April 2010

CRYPTOGRAPHIC PRIMITIVE

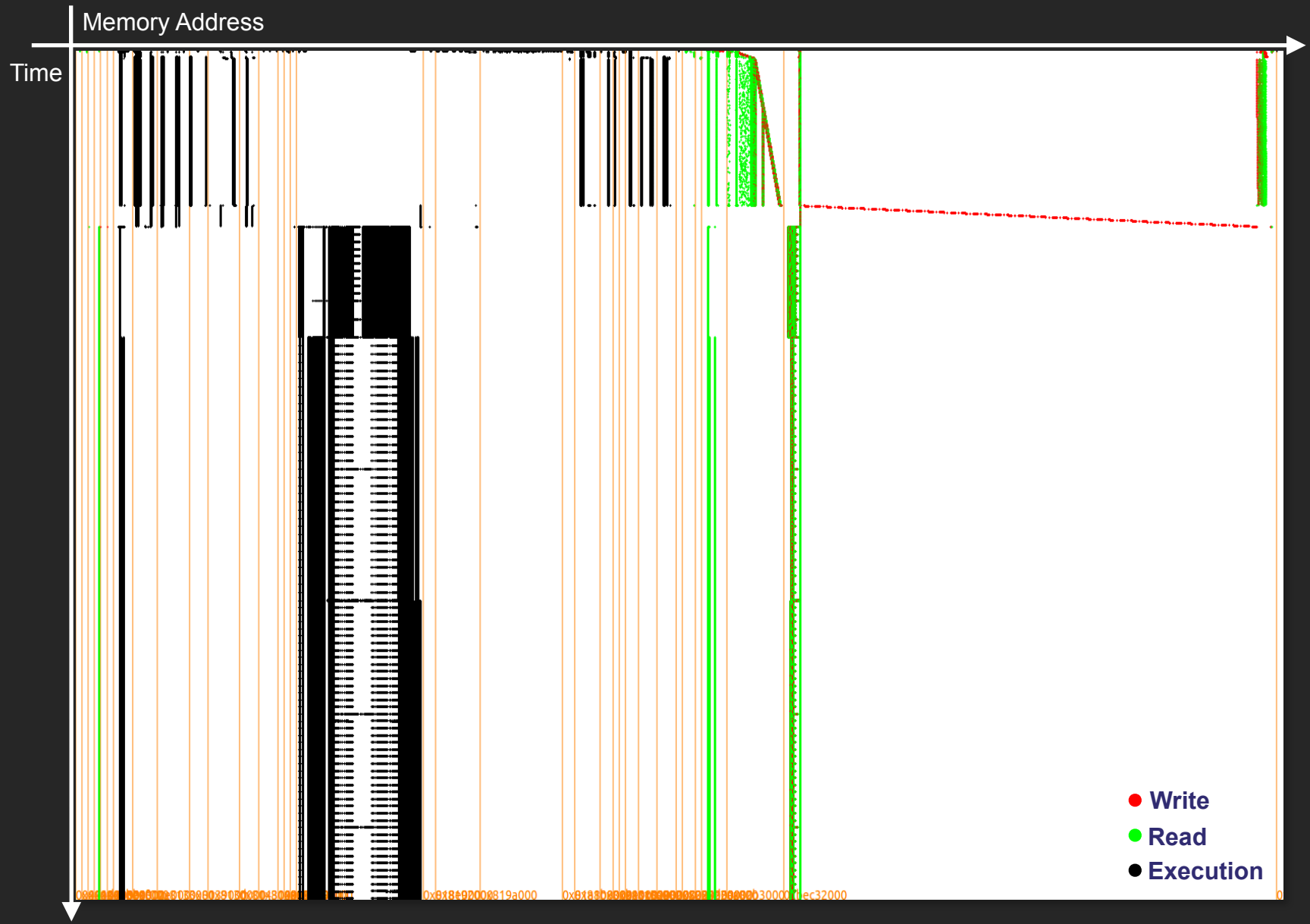


Typical example of a (hardware) power trace of an unprotected AES-128 implementation (one can observe the ten rounds)

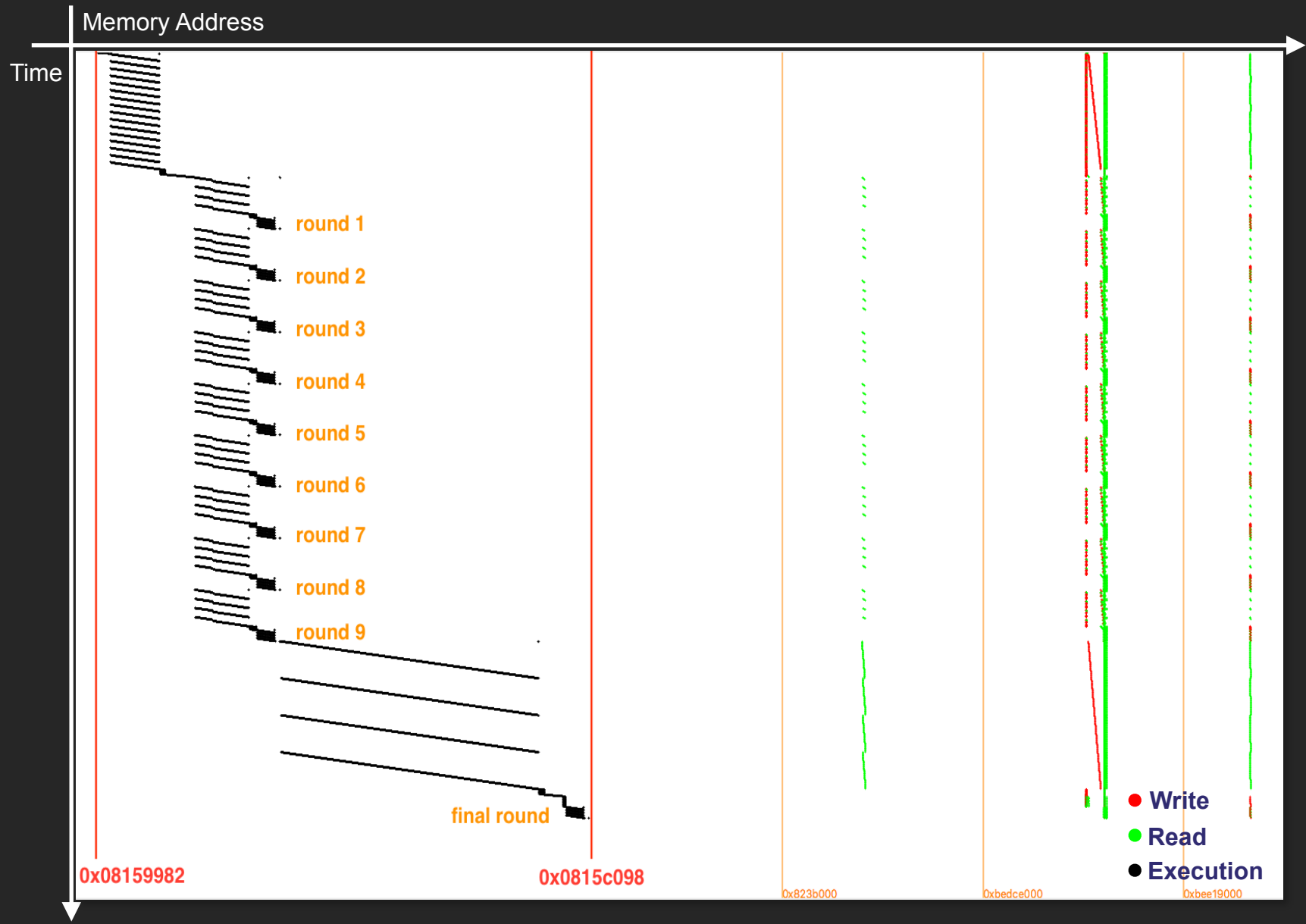
Reference : Kevin Meritt, “Differential Power Analysis attacks on AES”

SOFTWARE EXECUTION TRACE WITH DBI

VISUALIZED MAP OF OPEN-SOURCE WBAES-128



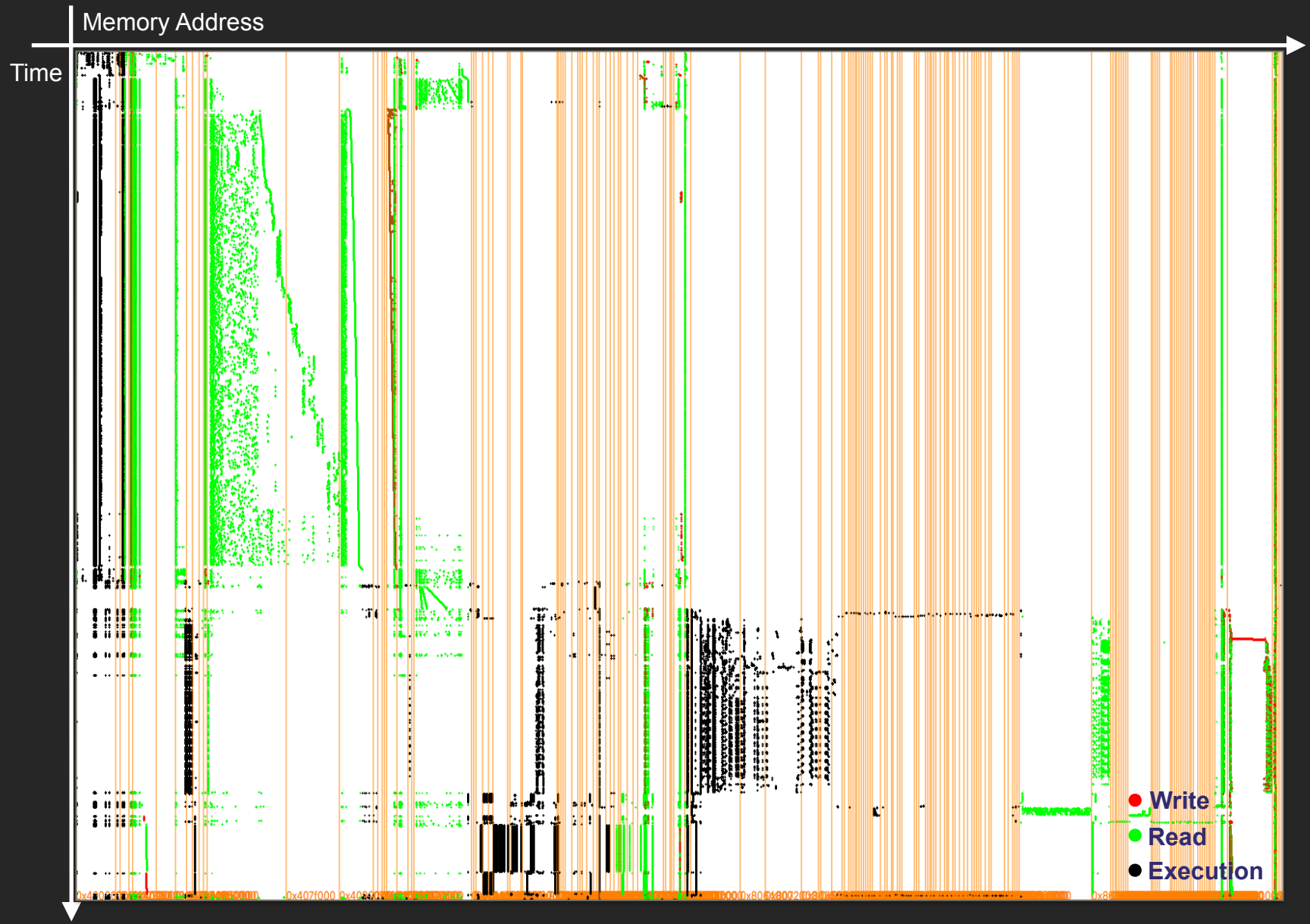
VISUALIZED MAP OF OPEN-SOURCE WBAES-128



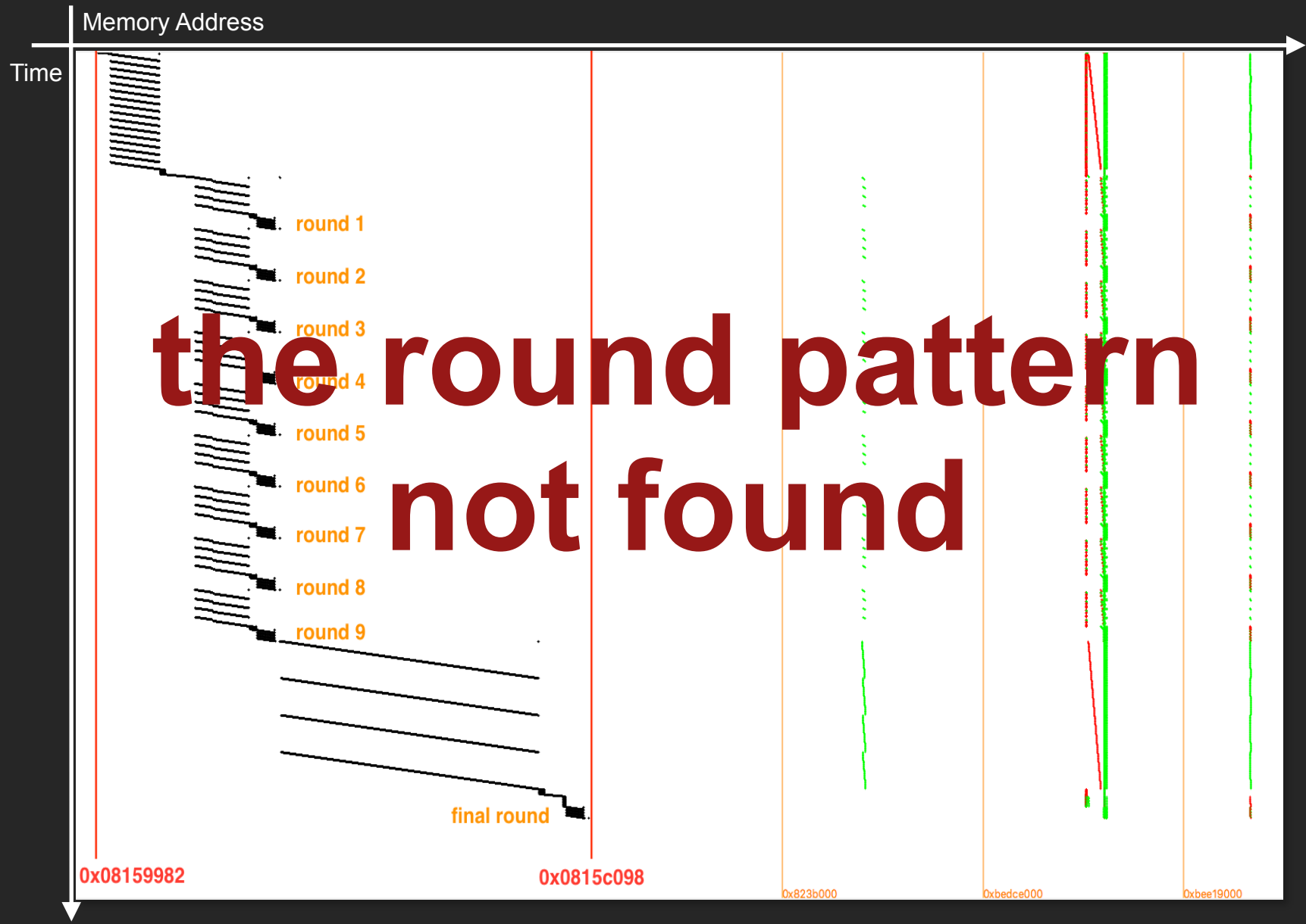
WBAES-128 KEY RECOVERY WITH CPA

	rank 1	rank 2	rank 3	rank 4	rank 5	rank 6	rank 7	rank 8	rank 9	rank 10
key 1	**0x2b** (2.00538)	0x07 (1.80379)	0xc1 (1.73824)	0xee (1.70429)	0x08 (1.697)	0x0c (1.69357)	0x72 (1.67844)	0x2c (1.67127)	0x81 (1.66722)	0x0a (1.65412)
key 2	**0x7e** (2.51091)	0x93 (1.74067)	0x06 (1.68867)	0x02 (1.68841)	0xa8 (1.67424)	0x54 (1.67341)	0x62 (1.67227)	0xb6 (1.66756)	0xe9 (1.66338)	0x86 (1.65473)
key 3	**0x15** (2.48606)	0xc8 (1.72634)	0xe4 (1.72553)	0x20 (1.70805)	0x92 (1.69391)	0x74 (1.68563)	0x1e (1.68499)	0x89 (1.67014)	0x43 (1.66724)	0xd6 (1.66098)
key 4	**0x16** (2.28568)	0xb5 (1.69677)	0x8f (1.68429)	0xbb (1.67537)	0x45 (1.66008)	0xe9 (1.64984)	0xcf (1.64378)	0xc2 (1.63327)	0x38 (1.61959)	0x6c (1.61645)
key 5	**0x28** (2.31281)	0x7d (1.69297)	0x5a (1.69283)	0xc9 (1.69053)	0x8e (1.68747)	0xfb (1.6714)	0xf3 (1.66702)	0x5b (1.6599)	0xaa (1.65702)	0x67 (1.65374)
key 6	**0xae** (2.5603)	0x69 (1.73705)	0x8a (1.73109)	0xf5 (1.67828)	0xc4 (1.67624)	0xfe (1.67155)	0xa0 (1.66758)	0xe5 (1.66617)	0x98 (1.6579)	0xb8 (1.65789)
key 7	**0xd2** (2.47133)	0xb4 (1.81599)	0x5f (1.75024)	0xe0 (1.74447)	0xa1 (1.73202)	0x5a (1.71384)	0x8b (1.70006)	0x0b (1.69626)	0xcd (1.68755)	0x68 (1.66165)
key 8	**0xa6** (1.80113)	0x4b (1.70776)	0x9d (1.69386)	0x2d (1.67404)	0x01 (1.6456)	0x0d (1.64192)	0xbf (1.63917)	0xea (1.63454)	0xe7 (1.62918)	0x3e (1.62822)
key 9	**0xab** (2.4717)	0x1d (1.7161)	0xc9 (1.71101)	0xe5 (1.70354)	0x7c (1.69538)	0x77 (1.68859)	0x2f (1.65904)	0xc4 (1.65721)	0xa1 (1.65257)	0x60 (1.64843)
key 10	**0xf7** (3.61634)	0xb8 (3.41278)	0xa4 (3.39504)	0x76 (3.38843)	0xc4 (3.38602)	0x17 (3.38253)	0xea (3.37196)	0xbe (3.35624)	0x8b (3.3548)	0x97 (3.33779)
key 11	**0x15** (2.02783)	0x16 (1.71121)	0xb6 (1.69905)	0x69 (1.69321)	0xbe (1.69316)	0x7b (1.69135)	0x41 (1.6857)	0x2f (1.67168)	0x91 (1.65783)	0x22 (1.65519)
key 12	**0x88** (2.8446)	0x3c (1.75204)	0xe4 (1.70494)	0x52 (1.66283)	0x79 (1.66221)	0x08 (1.64197)	0xb9 (1.62887)	0x40 (1.62375)	0x42 (1.61559)	0xb5 (1.60839)
key 13	**0x09** (1.88167)	0x96 (1.71604)	0xf2 (1.68481)	0xb4 (1.68284)	0x0d (1.67969)	0x6d (1.67959)	0x3f (1.67024)	0xaf (1.66383)	0x78 (1.66288)	0xe6 (1.66193)
key 14	**0xcf** (2.3232)	0x19 (1.76684)	0x9a (1.74075)	0x3d (1.72711)	0x03 (1.69709)	0xe2 (1.68798)	0x07 (1.68521)	0xd2 (1.67812)	0x6f (1.67621)	0xca (1.67311)
key 15	**0x4f** (2.11058)	0xd7 (1.80632)	0xce (1.71211)	0xad (1.71185)	0x45 (1.70138)	0x0e (1.69954)	0x62 (1.68237)	0x76 (1.67043)	0xc0 (1.66782)	0x48 (1.66293)
key 16	**0x3c** (2.22229)	0x97 (1.70098)	0xa7 (1.69181)	0x5d (1.68828)	0x02 (1.67629)	0xea (1.65554)	0x07 (1.65372)	0x7e (1.65026)	0xe6 (1.6479)	0x29 (1.64527)

VISUALIZED MAP OF SIMPLE-CIPHER

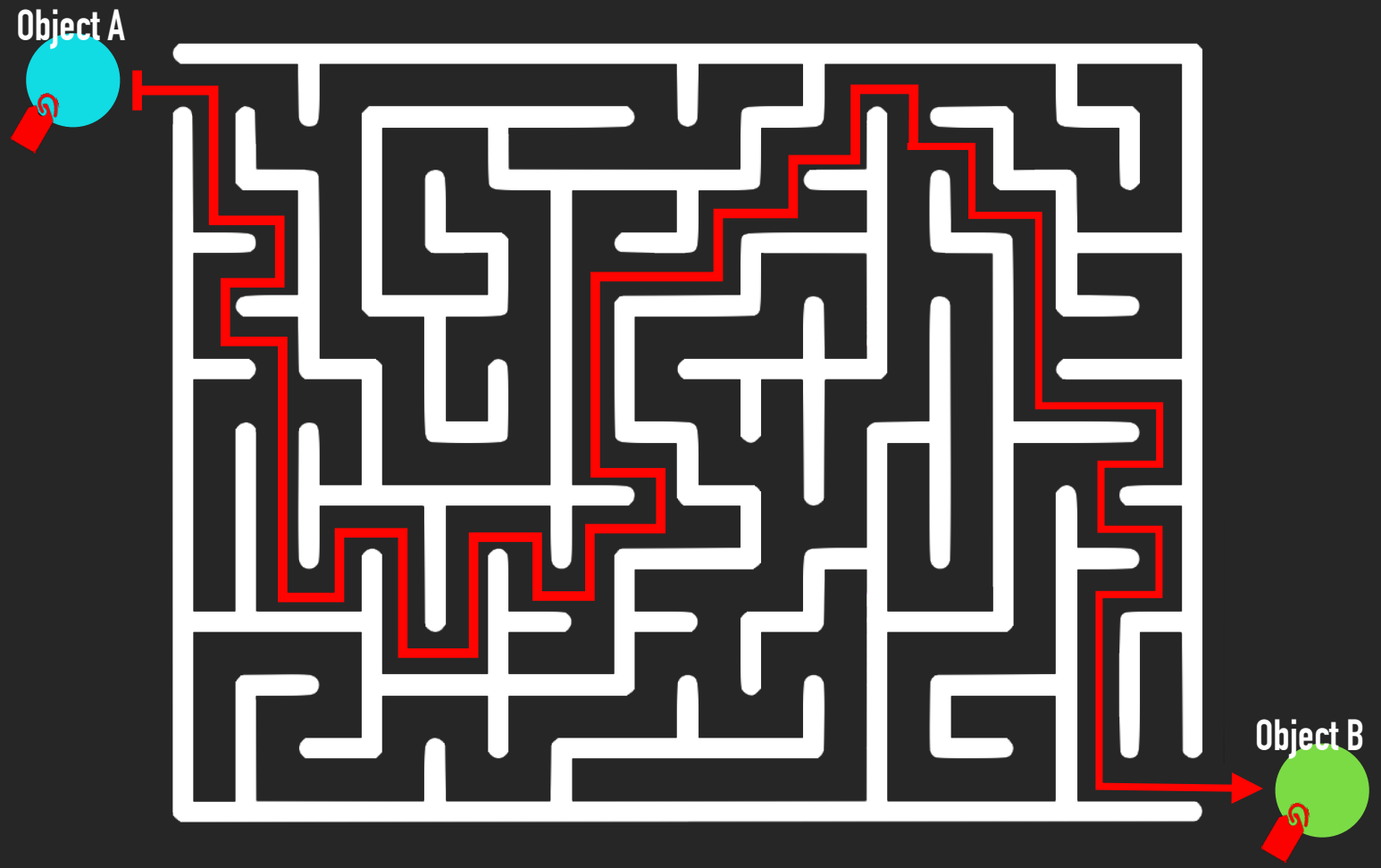


VISUALIZED MAP OF SIMPLE-CIPHER

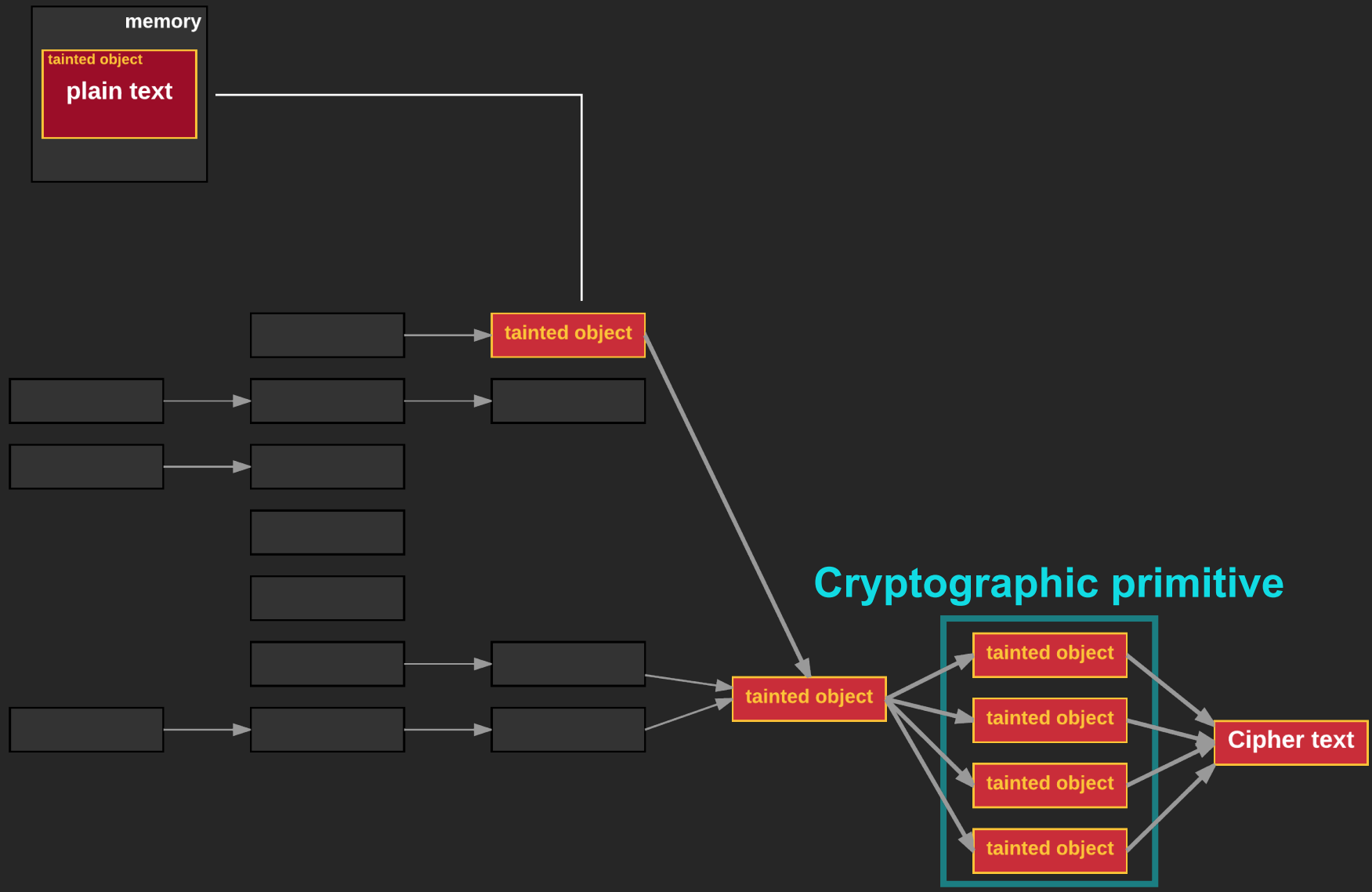


the round pattern not found

Taint analysis



TAINT ANALYSIS - PLAINTEXT TRACE

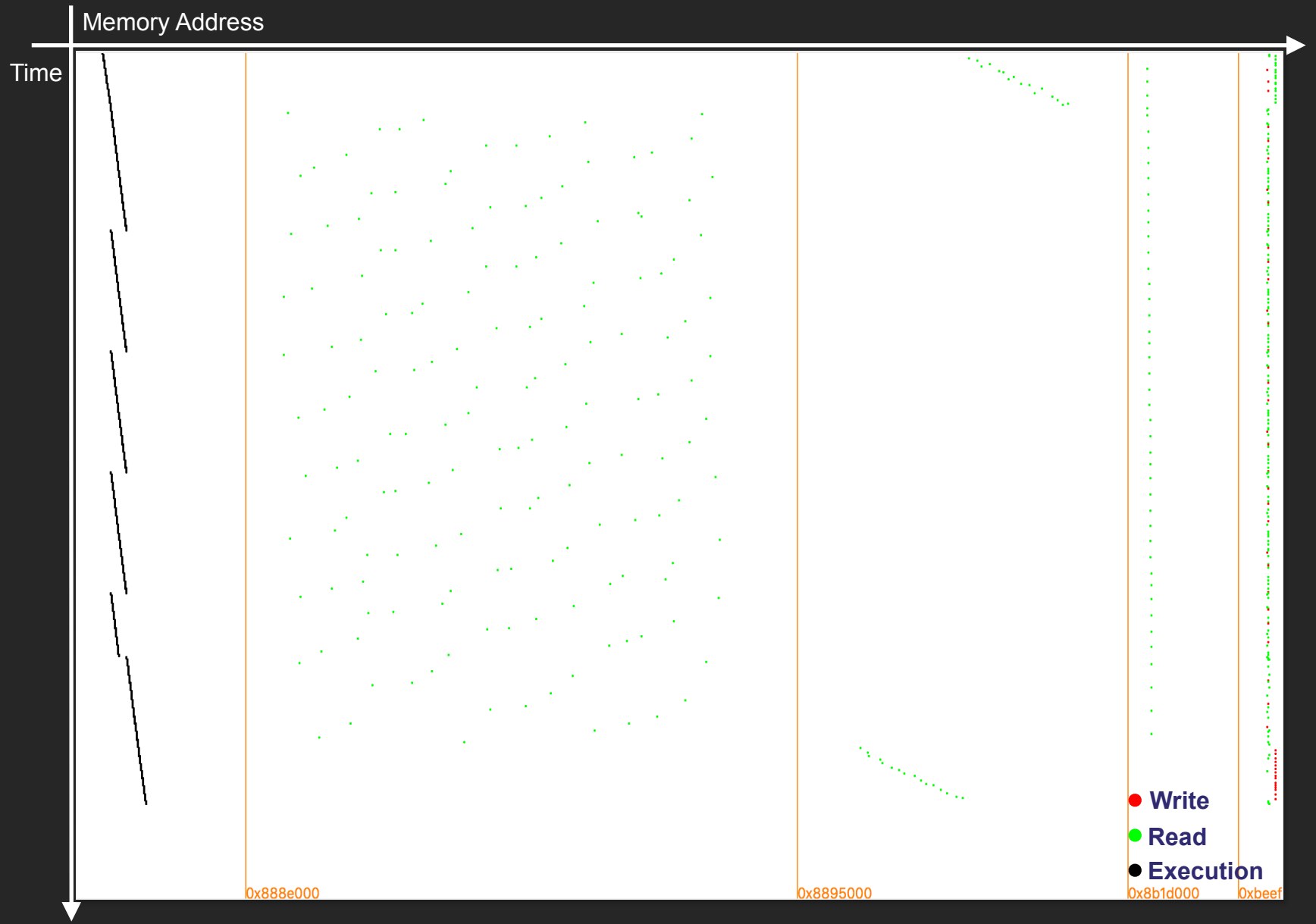


TAINT ANALYSIS FOR SIMPLE-CIPHER

```
0x4200986: (in /lib/i386-linux-gnu/libc-2.21.so)
0x42B34AE: (in /lib/i386-linux-gnu/libc-2.21.so)
0x42B34B3: (in /lib/i386-linux-gnu/libc-2.21.so)
0x42B34B8: (in /lib/i386-linux-gnu/libc-2.21.so)
0x42B34BD: (in /lib/i386-linux-gnu/libc-2.21.so)
…
0x8181ABA: (in Commercial-SimpleWB-AES)
0x8181AC4: (in Commercial-SimpleWB-AES)
0x8181ACC: (in Commercial-SimpleWB-AES)
0x8181AD0: (in Commercial-SimpleWB-AES)
0x8181AE0: (in Commercial-SimpleWB-AES)
0x8181AE4: (in Commercial-SimpleWB-AES)
0x8181AEE: (in Commercial-SimpleWB-AES)
0x8181AF2: (in Commercial-SimpleWB-AES)
0x8181B04: (in Commercial-SimpleWB-AES)
0x8181B08: (in Commercial-SimpleWB-AES)
0x8181B10: (in Commercial-SimpleWB-AES)
0x8181B14: (in Commercial-SimpleWB-AES)
0x8181B24: (in Commercial-SimpleWB-AES)
0x8181B28: (in Commercial-SimpleWB-AES)
0x8181B32: (in Commercial-SimpleWB-AES)
…
```

Cryptographic primitive

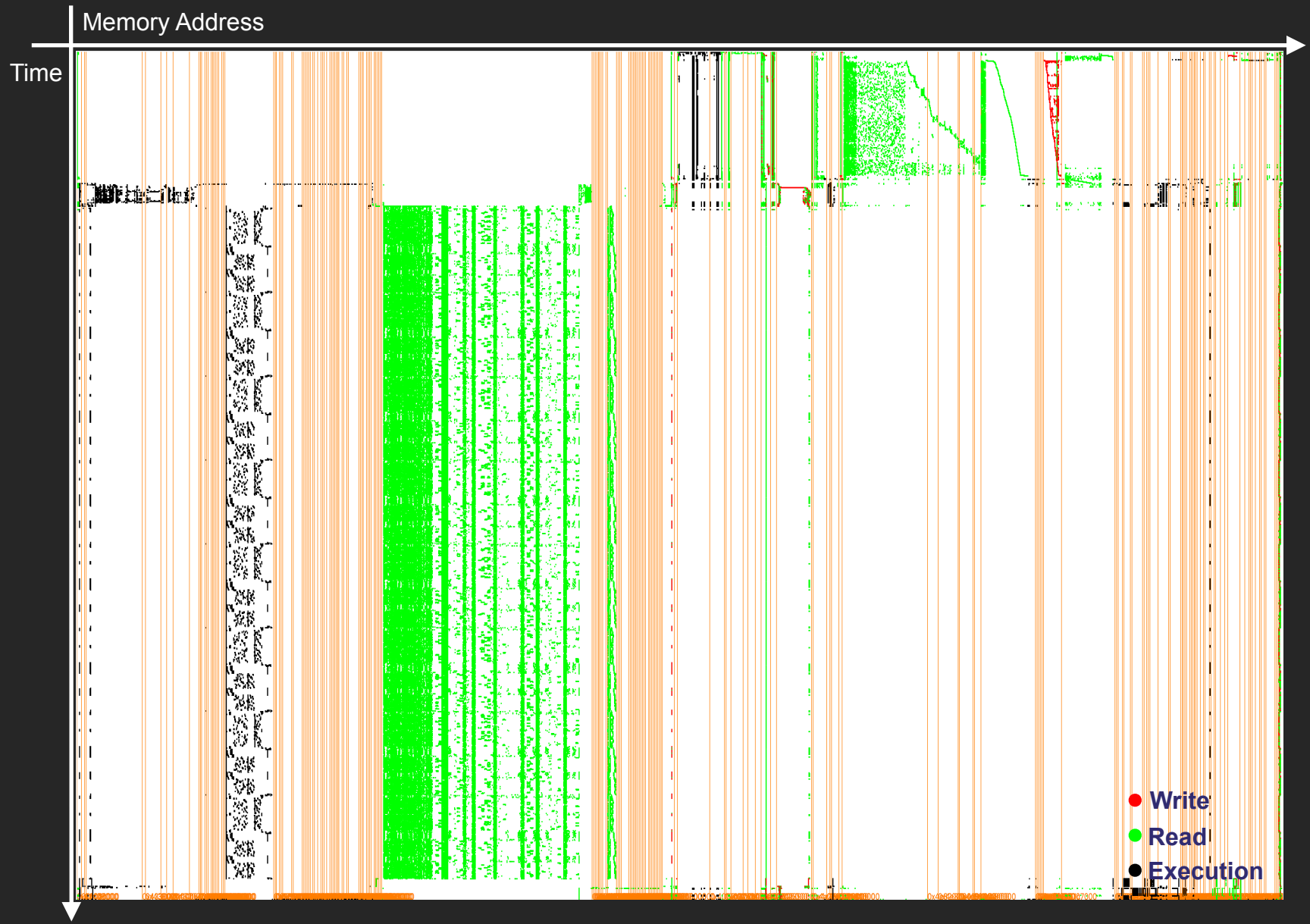
VISUALIZED MAP OF SIMPLE-CIPHER



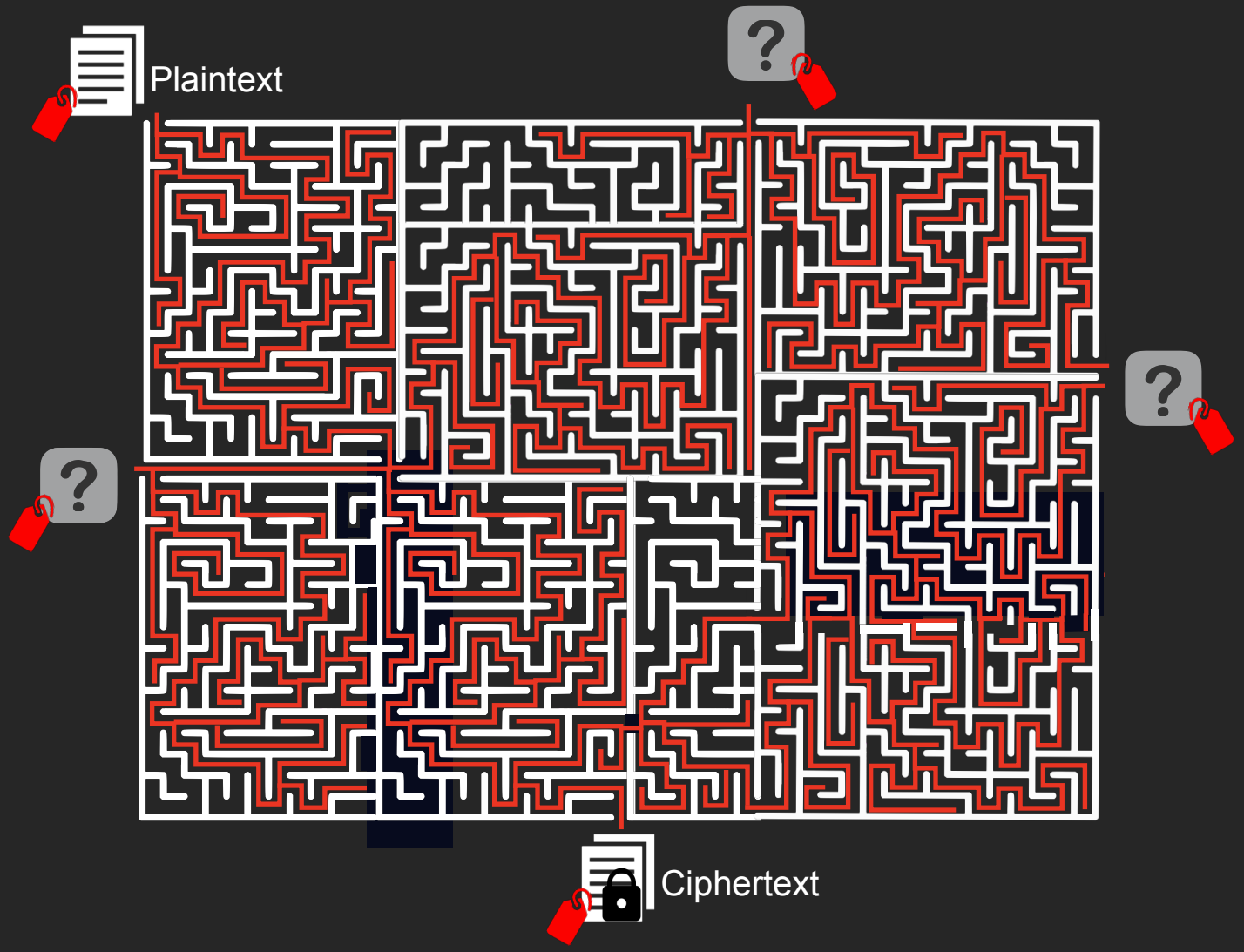
CPA ON THE SIMPLE-CIPHER

	rank 1	rank 2	rank 3	rank 4	rank 5	rank 6	rank 7	rank 8	rank 9	rank 10
key 1	0x4a (3.55009)	0x6b(3.38731)	0xbe(3.29755)	0x1e(3.29697)	0x2e(3.27285)	0x35(3.27269)	0x65(3.26551)	0x92(3.26066)	0x1d(3.25067)	0x58(3.24932)
key 2	0x32 (4.34452)	0xd0(3.35588)	0xe8(3.31946)	0xcc(3.30517)	0x78(3.29646)	0xdf(3.28295)	0x17(3.27702)	0x64(3.27235)	0x79(3.26674)	0x8a(3.26442)
key 3	0x4d (3.82747)	0xcc(3.31759)	0x23(3.29281)	0x91(3.28571)	0xb9(3.28026)	0xb3(3.27443)	0x4e(3.2596)	0xa2(3.25797)	0x8f(3.25309)	0x04(3.24456)
key 4	0x72 (3.64867)	0x38(3.35217)	0xdf(3.3211)	0x2f(3.31467)	0xae(3.30714)	0xa1(3.30303)	0xf8(3.28997)	0xd3(3.28245)	0x1b(3.26429)	0x8b(3.26017)
key 5	0x39 (4.1895)	0xc4(3.43936)	0xbb(3.32822)	0x8b(3.32537)	0x7c(3.31265)	0x8e(3.30741)	0x13(3.30152)	0x69(3.299)	0x9e(3.28805)	0x89(3.28379)
key 6	0x33 (3.62186)	0x2d(3.38423)	0xa8(3.31917)	0xb8(3.30849)	0x72(3.28712)	0x48(3.28305)	0x96(3.27886)	0x4d(3.27446)	0x23(3.27344)	0x9a(3.27234)
key 7	0x33 (4.26236)	0xda(3.30169)	0xc8(3.28577)	0x23(3.28246)	0x5f(3.26833)	0x17(3.26592)	0xd3(3.26428)	0xe6(3.26389)	0x64(3.25394)	0x85(3.24545)
key 8	0x6c (3.61456)	0x21(3.35728)	0xba(3.3402)	0xb3(3.32199)	0x65(3.29623)	0xaf(3.27848)	0x1f(3.27791)	0x61(3.27659)	0x44(3.27522)	0xc6(3.26389)
key 9	0x61 (4.19043)	0x8d(3.33732)	0x68(3.32288)	0x5f(3.30976)	0x1e(3.28015)	0xeb(3.27355)	0x96(3.26578)	0x13(3.26007)	0x0c(3.25348)	0xc0(3.25069)
key 10	0x54 (3.66626)	0x42(3.45208)	0xd6(3.38613)	0x5d(3.37338)	0x3f(3.3665)	0xbc(3.34422)	0x3a(3.33917)	0xed(3.33183)	0x2d(3.32099)	0x14(3.3029)
key 11	0x4e (3.71877)	0x74(3.32473)	0x39(3.32183)	0xee(3.30932)	0x52(3.30156)	0x68(3.27952)	0x7b(3.27918)	0x1a(3.27585)	0x30(3.25335)	0x16(3.25267)
key 12	0x6b (3.65183)	0xab(3.45354)	0x28(3.34018)	0xbc(3.33583)	0xc1(3.33411)	0x02(3.31736)	0x07(3.28696)	0x13(3.27714)	0x75(3.27475)	0xc0(3.26212)
key 13	0x32 (3.65053)	0x15(3.38309)	0x72(3.29484)	0xe9(3.28438)	0x88(3.28182)	0x52(3.25202)	0x95(3.24498)	0x6c(3.24336)	0x2d(3.24149)	0xc7(3.22936)
key 14	0x4d (3.4734)	0x7d(3.31076)	0xcd(3.30883)	0x8e(3.30059)	0x5f(3.28006)	0x0b(3.27518)	0x10(3.26867)	0xee(3.26289)	0x7b(3.2615)	0x1a(3.24988)
key 15	0x4a (3.65855)	0x0a(3.30734)	0x89(3.29788)	0xaf(3.29663)	0xf0(3.2857)	0xf1(3.28402)	0xcd(3.26806)	0x48(3.26561)	0xc8(3.26545)	0x87(3.25869)
key 16	0x30 (4.20028)	0xc0(3.36153)	0xce(3.3276)	0x2c(3.32361)	0xe9(3.30428)	0x7b(3.29937)	0x8f(3.29511)	0x4e(3.29502)	0x1c(3.28909)	0xd6(3.27938)

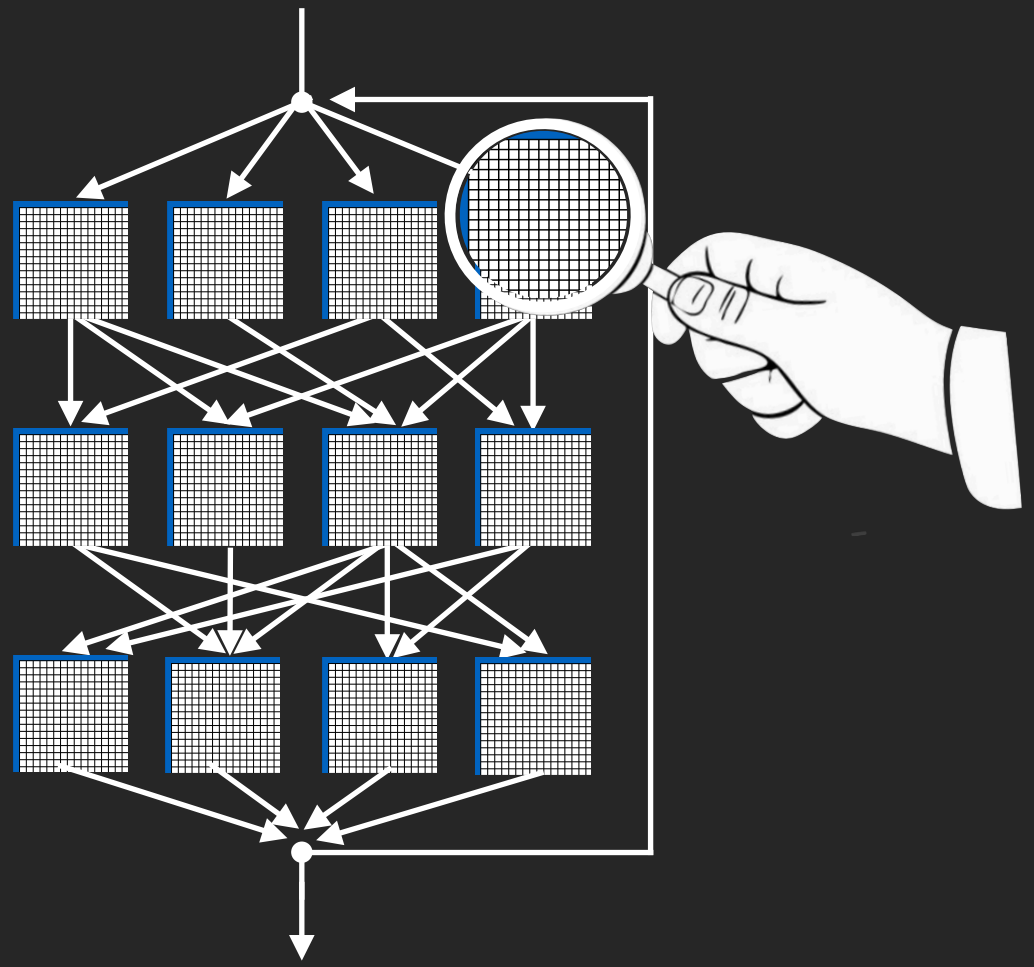
VISUALIZED MAP OF COMPLEX-CIPHER

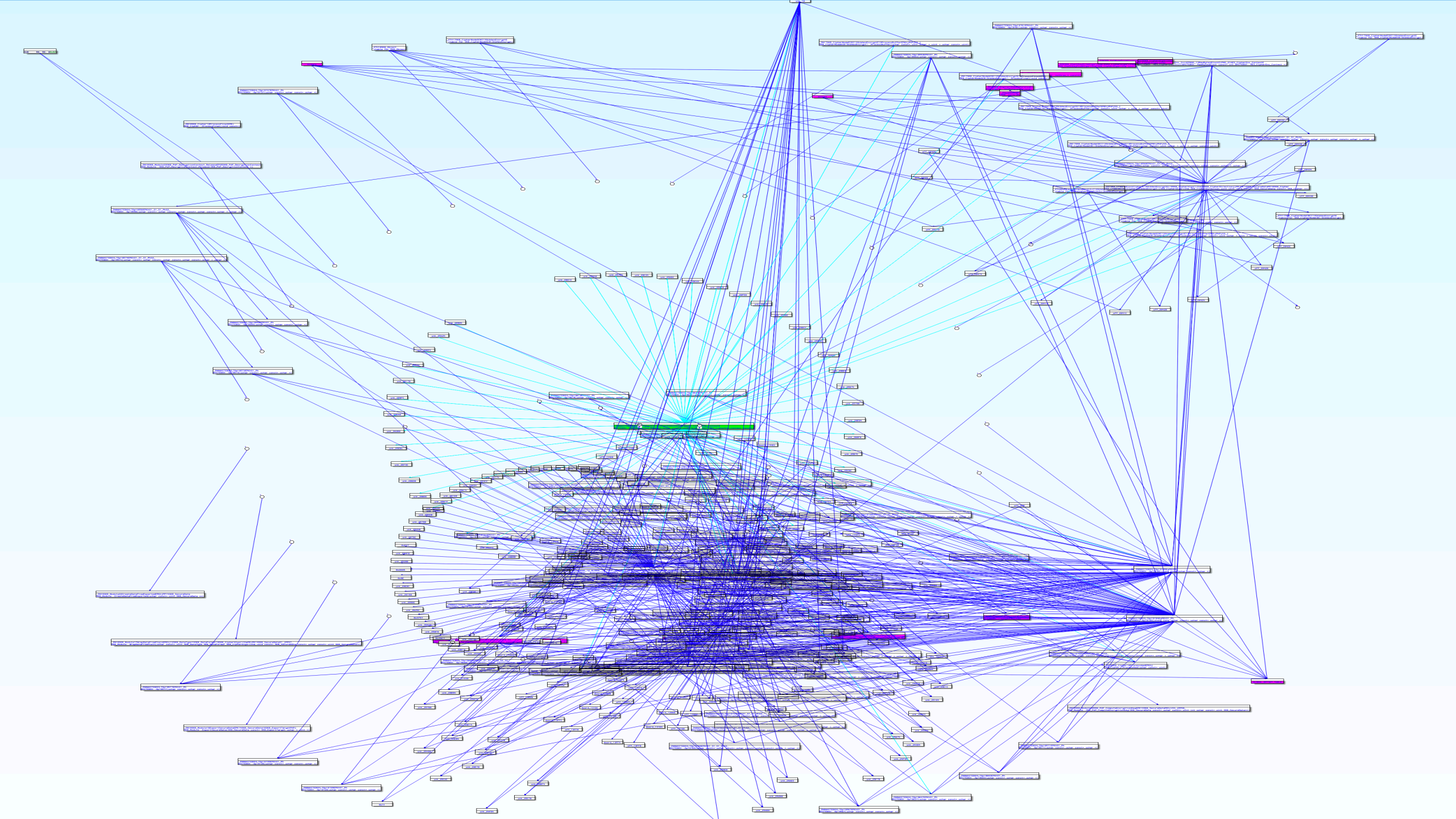


PLAINTEXT TRACE ON THE COMPLEX-CIPHER

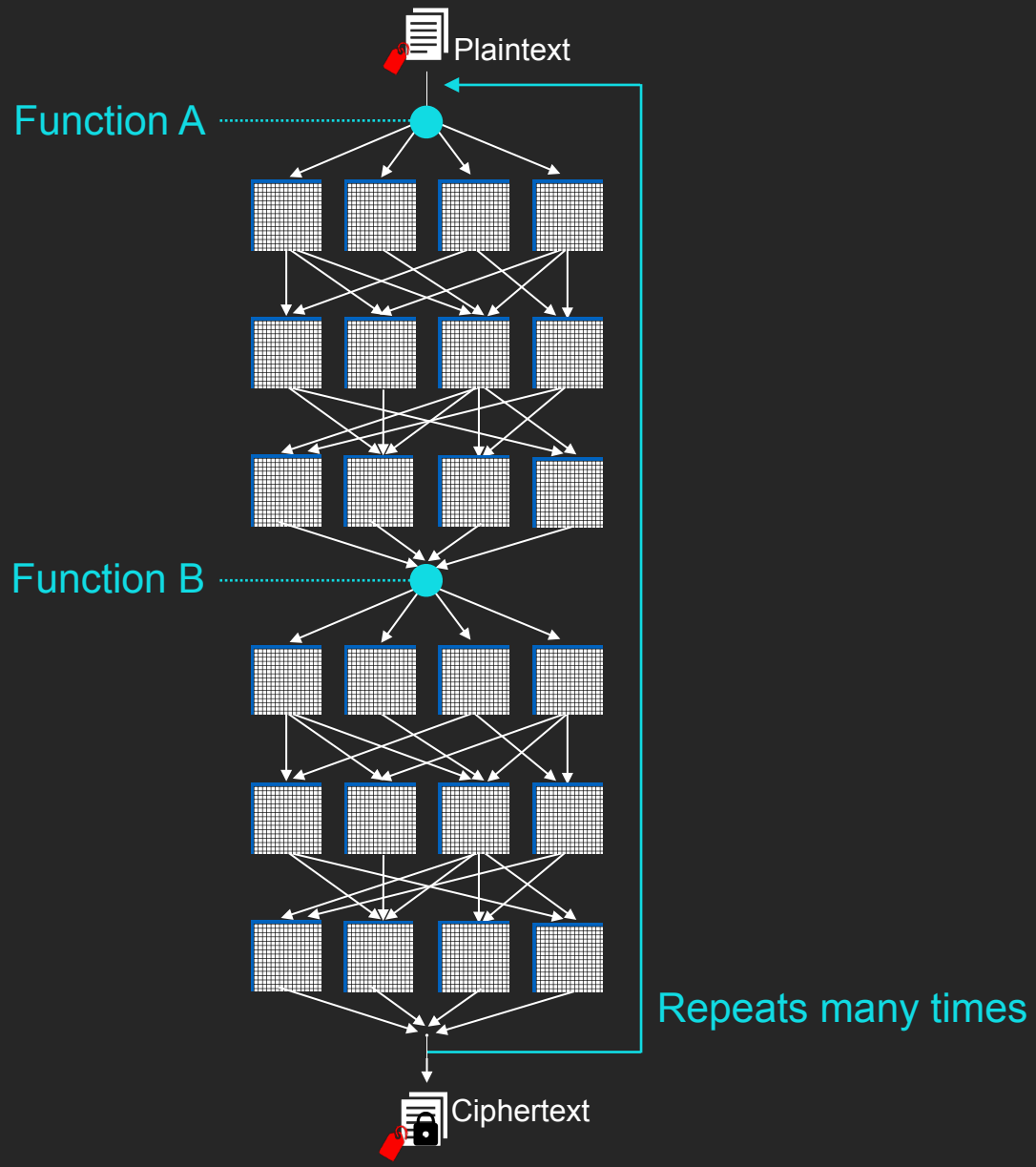


Reverse engineering for complex cipher

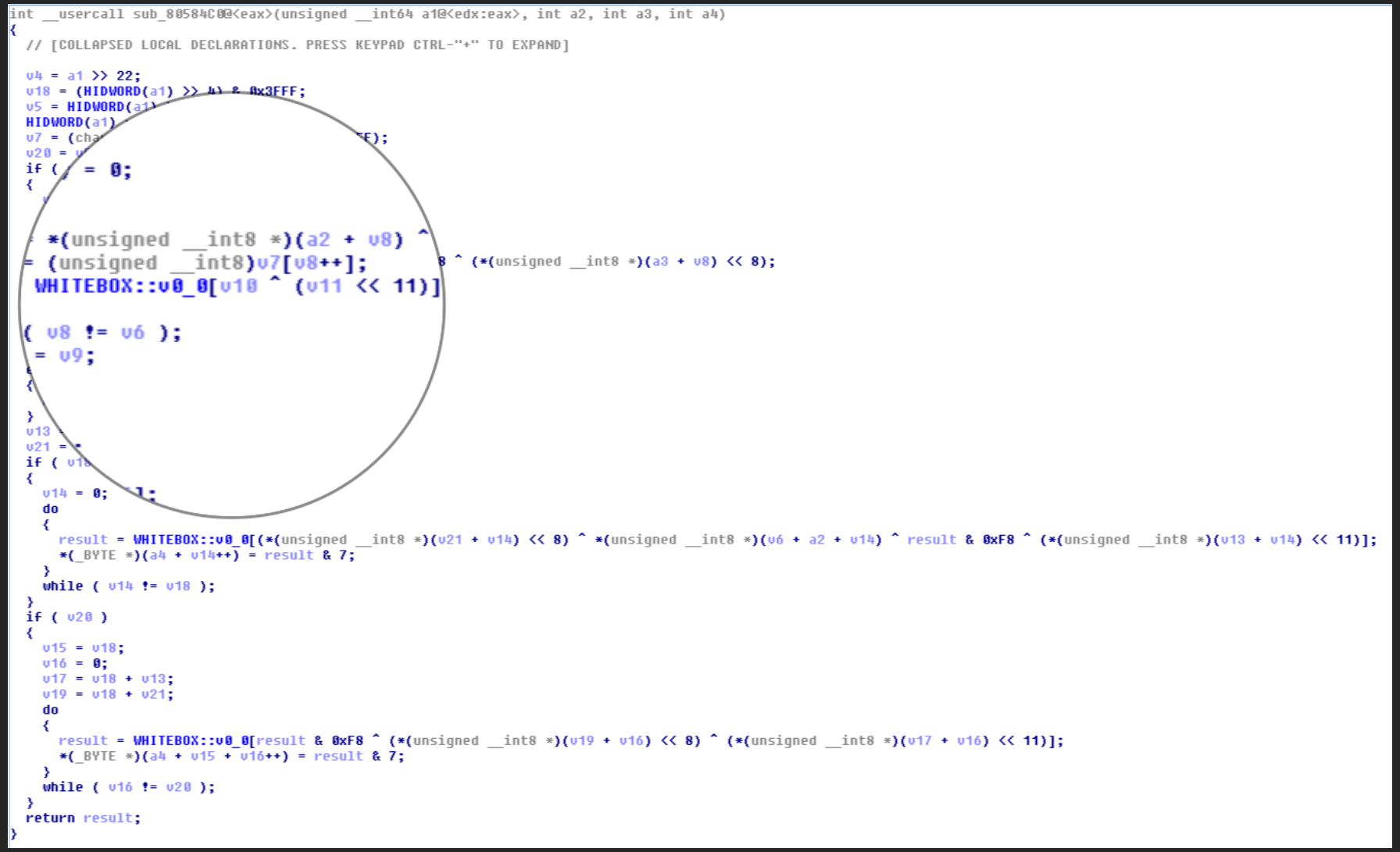




COMPLEX-CIPHER CONTROL FLOW



WHITE-BOX TABLE IN COMPLEX-CIPHER



```
int __usercall sub_80584C0@<eax>(unsigned __int64 a1@<edx:eax>, int a2, int a3, int a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v4 = a1 >> 22;
  v18 = (HIDWORD(a1) >> 4) & 0x3FFF;
  v5 = HIDWORD(a1)
  HIDWORD(a1)
  v7 = (cha                              FF);
  v20 = v
  if ( v  = 0;
  {

    *(unsigned __int8 *)(a2 + v8) ^
  = (unsigned __int8)v7[v8++];                8 ^ (*(unsigned __int8 *)(a3 + v8) << 8);
   WHITEBOX::v0_0[v10 ^ (v11 << 11)]

  ( v8 != v6 );
   = v9;
  {
  {

  }
  v13
  v21 =
  if ( v1
  {
    v14 = 0;
    do
    {
      result = WHITEBOX::v0_0[(*(unsigned __int8 *)(v21 + v14) << 8) ^ *(unsigned __int8 *)(v6 + a2 + v14) ^ result & 0xF8 ^ (*(unsigned __int8 *)(v13 + v14) << 11)];
      *(_BYTE *)(a4 + v14++) = result & 7;
    }
    while ( v14 != v18 );
  }
  if ( v20 )
  {
    v15 = v18;
    v16 = 0;
    v17 = v18 + v13;
    v19 = v18 + v21;
    do
    {
      result = WHITEBOX::v0_0[result & 0xF8 ^ (*(unsigned __int8 *)(v19 + v16) << 8) ^ (*(unsigned __int8 *)(v17 + v16) << 11)];
      *(_BYTE *)(a4 + v15 + v16++) = result & 7;
    }
    while ( v16 != v20 );
  }
  return result;
}
```

FAULT ANALYSIS

- Modify intermediate data
- Record changes to the output
- Compare incorrect result and correct result

DIFFERENTIAL FAULT ANALYSIS(DFA)

After ShiftRow9

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Fault injected '1E'

99	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

After Mixcolumns

7B	40	43	4C
29	D4	70	9F
8A	E4	3A	42
CF	A5	A6	BC

$\oplus$ K_9

AC	19	28	57
77	FA	D1	5C
66	D	29	00
F3	21	41	6E

After AddRoundKey9

D7	59	8B	1B
5E	2E	A1	C3
EC	38	13	42
3C	84	E7	D2

After SubBytes10

0E	C	3D	AF
58	31	32	2E
CE	07	7D	2C
EB	5F	94	B5

After ShiftRows10

0E	C	3D	AF
31	32	2E	58
7D	2C	CE	07
B5	EB	5F	94

$\oplus$ K_{10}

D0	C9	E1	B6
14	EE	3F	63
F9	25	0C	0C
A8	89	C8	A6

$=$

Output with faults

DE	02	D	19
25	D	11	3B
84	09	C2	0B
1D	62	97	32

Output without fault

39	02	D	19
25	D	11	6A
84	09	85	0B
1D	FB	97	32

- Input = '3243F6A8885A308D313198A2E0370734'
- Cipher Key ='2B7E151628AED2A6ABF7158809CF4F3C'
- Output= '3925841D02DC09FBDC118597196A0B32'

Reference : P. Dusart, G. Letourneux and O. Vivolo, "Differential Fault Analysis on A.E.S.," Cryptology ePrint Archive of IACR, No. 010, 2003

DIFFERENTIAL FAULT ANALYSIS(DFA)

After ShiftRow9

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Fault injected '1E'

99	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

After Mixcolumns

7B	40	43	4C
29	D4	70	9F
8A	E4	3A	42
CF	A5	A6	BC

$\oplus$ K_9

AC	19	28	57
77	FA	D1	5C
66	D	29	00
F3	21	41	6E

After AddRoundKey9

D7	59	8B	1B
5E	2E	A1	C3
EC	38	13	42
3C	84	E7	D2

After SubBytes10

0E	C	3D	AF
58	31	32	2E
CE	07	7D	2C
EB	5F	94	B5

After ShiftRows10

0E	C	3D	AF
31	32	2E	58
7D	2C	CE	07
B5	EB	5F	94

$\oplus$ K_{10}

D0	C9	E1	B6
14	EE	3F	63
F9	25	0C	0C
A8	89	C8	A6

$=$

Output with faults

DE	02	D	19
25	D	11	3B
84	09	C2	0B
1D	62	97	32

$\oplus$

Output without fault

39	02	D	19
25	D	11	6A
84	09	85	0B
1D	FB	97	32

$=$

Error

E7	00	00	00
00	00	00	51
00	00	47	00
00	99	00	00

- Input = '3243F6A8885A308D313198A2E0370734'
- Cipher Key ='2B7E151628AED2A6ABF7158809CF4F3C'
- Output= '3925841D02DC09FBDC118597196A0B32'

Reference : P. Dusart, G. Letourneux and O. Vivolo, "Differential Fault Analysis on A.E.S.," Cryptology ePrint Archive of IACR, No. 010, 2003

DIFFERENTIAL FAULT ANALYSIS(DFA)

After ShiftRow9

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

Fault injected '1E'

99	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

After Mixcolumns

7B	40	43	4C
29	D4	70	9F
8A	E4	3A	42
CF	A5	A6	BC

$\oplus$ K_9

AC	19	28	57
77	FA	D1	5C
66	D	29	00
F3	21	41	6E

After AddRoundKey9

D7	59	8B	1B
5E	2E	A1	C3
EC	38	13	42
3C	84	E7	D2

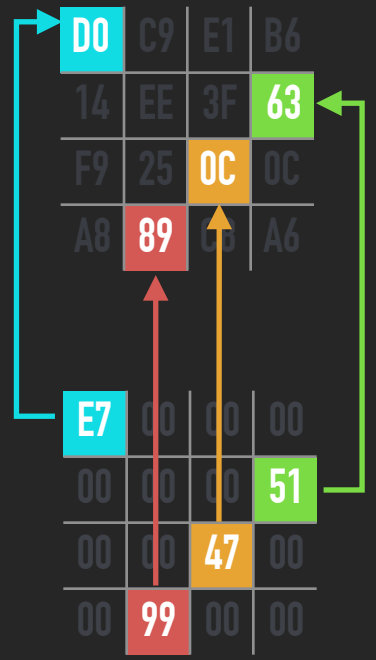
After SubBytes10

0E	C	3D	AF
58	31	32	2E
CE	07	7D	2C
EB	5F	94	B5

After ShiftRows10

0E	C	3D	AF
31	32	2E	58
7D	2C	CE	07
B5	EB	5F	94

$\oplus$ K_{10}



Error

$=$

Output with faults

DE	02	D	19
25	D	11	3B
84	09	C2	0B
1D	62	97	32

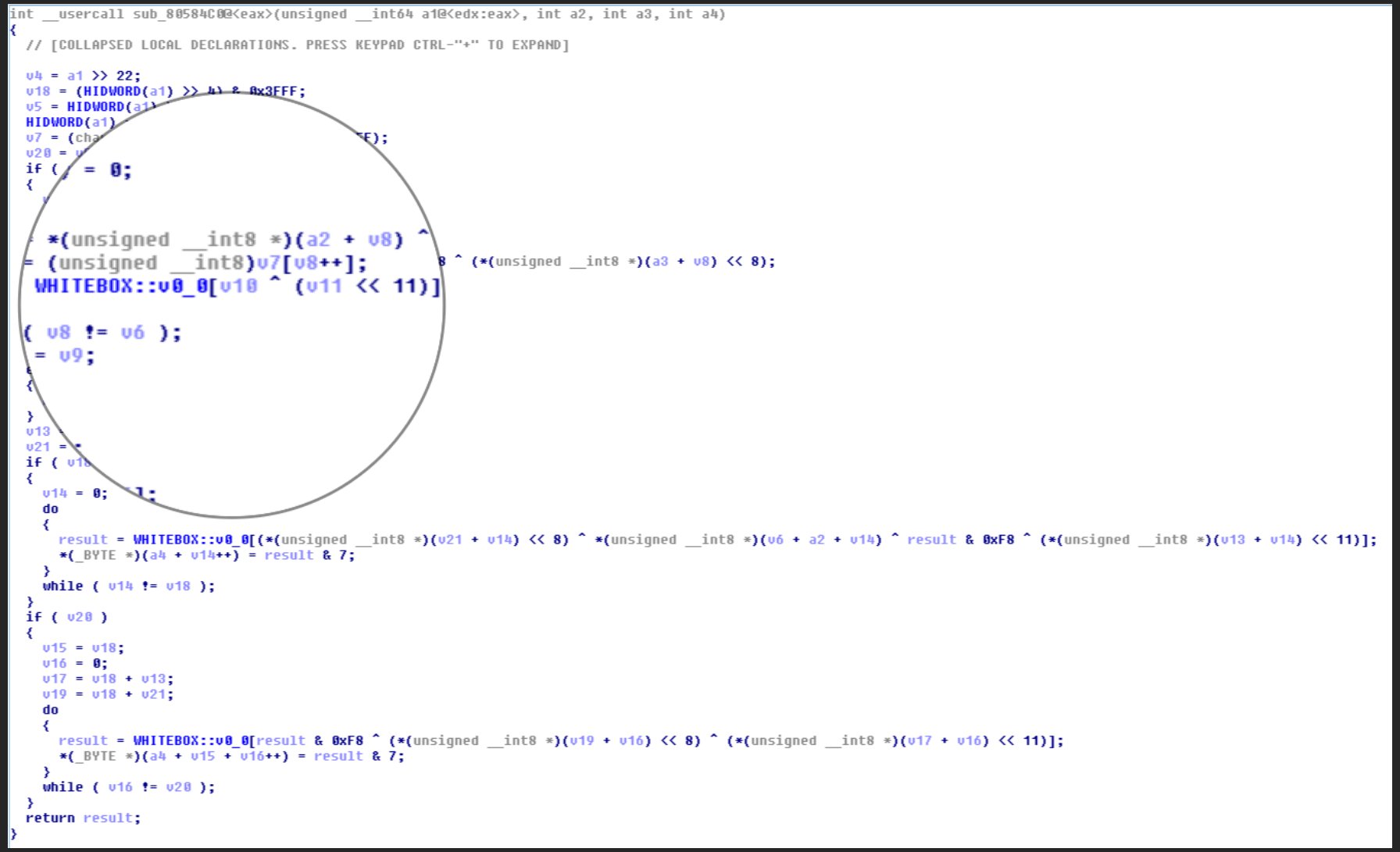
Output without fault

39	02	D	19
25	D	11	6A
84	09	85	0B
1D	FB	97	32

- Input = '3243F6A8885A308D313198A2E0370734'
- Cipher Key ='2B7E151628AED2A6ABF7158809CF4F3C'
- Output= '3925841D02DC09FBDC118597196A0B32'

Reference : P. Dusart, G. Letourneux and O. Vivolo, "Differential Fault Analysis on A.E.S.," Cryptology ePrint Archive of IACR, No. 010, 2003

WHITE-BOX TABLE IN COMPLEX-CIPHER



```
int __usercall sub_80584C0@<eax>(unsigned __int64 a1@<edx:eax>, int a2, int a3, int a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v4 = a1 >> 22;
  v18 = (HIDWORD(a1) >> 4) & 0x3FFF;
  v5 = HIDWORD(a1)
  HIDWORD(a1)
  v7 = (cha                              FF);
  v20 = v
  if ( v  = 0;
  {

    *(unsigned __int8 *)(a2 + v8) ^
  = (unsigned __int8)v7[v8++];                8 ^ (*(unsigned __int8 *)(a3 + v8) << 8);
   WHITEBOX::v0_0[v10 ^ (v11 << 11)]

  ( v8 != v6 );
   = v9;
  {
  {

  }
  v13
  v21 =
  if ( v1
  {
    v14 = 0;
    do
    {
      result = WHITEBOX::v0_0[(*(unsigned __int8 *)(v21 + v14) << 8) ^ *(unsigned __int8 *)(v6 + a2 + v14) ^ result & 0xF8 ^ (*(unsigned __int8 *)(v13 + v14) << 11)];
      *(_BYTE *)(a4 + v14++) = result & 7;
    }
    while ( v14 != v18 );
  }
  if ( v20 )
  {
    v15 = v18;
    v16 = 0;
    v17 = v18 + v13;
    v19 = v18 + v21;
    do
    {
      result = WHITEBOX::v0_0[result & 0xF8 ^ (*(unsigned __int8 *)(v19 + v16) << 8) ^ (*(unsigned __int8 *)(v17 + v16) << 11)];
      *(_BYTE *)(a4 + v15 + v16++) = result & 7;
    }
    while ( v16 != v20 );
  }
  return result;
}
```

WHITE-BOX TABLE IN COMPLEX-CIPHER

```
.rodata:085B2000                    public _ZN8WHITEBOX4v0_0E
.rodata:085B2000 ; unsigned __int8 WHITEBOX::v0_0[131072]
.rodata:085B2000 _ZN8WHITEBOX4v0_0E db 0BDh, 62h, 4Ch, 0BFh, 66h, 60h, 0F9h, 0DBh, 39h, 2 dup(38h)
.rodata:085B2000                                        ; DATA XREF: sub_8056250+49↑o
.rodata:085B2000                                        ; sub_8056250+A3↑o ...
.rodata:085B2000                    db 19h, 18h, 19h, 18h, 39h, 91h, 0E7h, 90h, 9, 6Fh, 6Ah
.rodata:085B2000                    db 90h, 2 dup(6Ah), 90h, 6Fh, 0E2h, 8, 9, 0E7h, 91h, 39h
.rodata:085B2000                    db 2 dup(38h), 19h, 18h, 19h, 18h, 39h, 0DEh, 0DBh, 67h
.rodata:085B2000                    db 62h, 48h, 0DCh, 0FDh, 49h, 3Ah, 1Fh, 3Fh, 1Ah, 3Fh
.rodata:085B2000                    db 3Ah, 3Fh, 3Ah, 91h, 0E7h, 90h, 9, 6Fh, 6Ah, 90h, 6Ah
.rodata:085B2000                    db 5Bh, 0CCh, 0C2h, 0C8h, 59h, 0CFh, 8Eh, 5, 0CCh, 0CDh
.rodata:085B2000                    db 0B3h, 59h, 0CFh, 8Eh, 88h, 0C2h, 5Dh, 5Eh, 0B4h, 0CFh
.rodata:085B2000                    db 0CAh, 0C8h, 89h, 83h, 0DDh, 0DEh, 61h, 67h, 4Ah, 53h
.rodata:085B2000                    db 0FCh, 40h, 0C2h, 0B3h, 0CFh, 8Eh, 0C0h, 4, 0CDh, 81h
.rodata:085B2000                    db 3Ah, 1Fh, 3Fh, 1Ah, 3Fh, 3Ah, 3Fh, 3Ah, 96h, 0E0h, 6Dh
.rodata:085B2000                    db 0Eh, 68h, 69h, 6Dh, 2 dup(69h), 6Dh, 68h, 0E1h, 0E5h
.rodata:085B2000                    db 0Eh, 0E0h, 96h, 0D3h, 0FCh, 0DEh, 0B8h, 61h, 67h, 62h
.rodata:085B2000                    db 0DDh, 4Ch, 4Dh, 60h, 0F9h, 4Fh, 4Ah, 0FBh, 4Eh, 39h
.rodata:085B2000                    db 18h, 38h, 19h, 38h, 39h, 38h, 2 dup(39h), 18h, 38h
.rodata:085B2000                    db 19h, 38h, 39h, 38h, 39h, 3Ah, 2 dup(3Fh), 1Ah, 1Fh
.rodata:085B2000                    db 1Ah, 1Fh, 3Ah, 6Ah, 90h, 6Fh, 0E2h, 8, 9, 0E7h, 91h
.rodata:085B2000                    db 0FCh, 0FDh, 0DBh, 61h, 67h, 62h, 60h, 0DEh, 5, 0C2h
.rodata:085B2000                    db 59h, 0CFh, 0C6h, 83h, 0CCh, 0, 62h, 0FBh, 4Dh, 66h
.rodata:085B2000                    db 60h, 0F9h, 67h, 4Ch, 0DBh, 4Ch, 66h, 60h, 0D9h, 45h
.rodata:085B2000                    db 62h, 47h, 83h, 0B4h, 0CAh, 0C8h, 0C1h, 85h, 5Eh, 7
.rodata:085B2000                    db 5Eh, 5Bh, 0B5h, 0CAh, 0C8h, 89h, 8Fh, 0B4h, 3Ah, 2 dup(3Fh)
.rodata:085B2000                    db 1Ah, 1Fh, 1Ah, 1Fh, 3Ah, 69h, 6Dh, 68h, 0E1h, 0E5h
.rodata:085B2000                    db 0Eh, 0E0h, 2 dup(96h), 0E0h, 6Dh, 0Eh, 68h, 69h, 6Dh
.rodata:085B2000                    db 69h, 0B4h, 0B5h, 0C8h, 89h, 0C7h, 86h, 5Bh, 2, 0BCh
.rodata:085B2000                    db 0BDh, 0DBh, 0D1h, 0BFh, 66h, 60h, 4Ah, 3 dup(39h), 4 dup(19h)
.rodata:085B2000                    db 39h, 91h, 0E2h, 91h, 9, 2 dup(6Ah), 91h, 2 dup(6Ah)
.rodata:085B2000                    db 91h, 6Ah, 0E2h, 2 dup(9), 0E2h, 91h, 3 dup(39h), 4 dup(19h)
.rodata:085B2000                    db 39h, 0DDh, 0DEh, 61h, 67h, 4Ah, 53h, 0FCh, 40h, 3Ah
.rodata:085B2000                    db 1Ah, 3Ah, 1Ah, 4 dup(3Ah), 91h, 0E2h, 91h, 9, 2 dup(6Ah)
.rodata:085B2000                    db 91h, 6Ah, 0CAh, 5Bh, 5, 0CEh, 0C8h, 59h, 0CFh, 4, 5Bh
.rodata:085B2000                    db 0CCh, 0C2h, 0C8h, 59h, 0CFh, 8Eh, 5, 5Ch, 5Dh, 83h
.rodata:085B2000                    db 0C9h, 0CFh, 0CAh, 0C8h, 86h, 0DCh, 0DDh, 0B8h, 61h
.rodata:085B2000                    db 4Fh, 56h, 0D3h, 42h, 5, 0C2h, 59h, 0CFh, 0C6h, 83h
.rodata:085B2000                    db 0CCh, 0, 3Ah, 1Ah, 3Ah, 1Ah, 4 dup(3Ah), 96h, 0E1h
.rodata:085B2000                    db 96h, 0Eh, 2 dup(69h), 96h, 2 dup(69h), 96h, 69h, 0E1h
.rodata:085B2000                    db 2 dup(0Eh), 0E1h, 96h, 0D6h, 0D3h, 0DDh, 0BAh, 0B8h
.rodata:085B2000                    db 61h, 67h, 0DCh, 0DBh, 4Ch, 66h, 60h, 0D9h, 45h, 62h
.rodata:085B2000                    db 47h, 39h, 19h, 39h, 19h, 5 dup(39h), 19h, 39h, 19h
.rodata:085B2000                    db 4 dup(39h), 3 dup(3Ah), 4 dup(1Ah), 3Ah, 6Ah, 91h, 6Ah
.rodata:085B2000                    db 0E2h, 2 dup(9), 0E2h, 91h, 0D3h, 0FCh, 0DEh, 0B8h, 61h
.rodata:085B2000                    db 67h, 62h, 0DDh, 4, 5, 0C8h, 59h, 7, 2, 5Bh, 6, 0BDh
.rodata:085B2000                    db 62h, 4Ch, 0BFh, 66h, 60h, 0F9h, 0DBh, 4Ah, 0DBh, 0BFh
```

WHITE-BOX TABLE IN COMPLEX-CIPHER

1byte flip;0xBD→0x34

FAULTY CIPHERTEXT

```
h2spice@ubuntu:~/Documents/WBC/Commercial/DFA/complexWB-AES$
./complexWB-AES testtesttesttest
6CB721A5633DFD7F94A6474524789026 ← normal cipher-text

h2spice@ubuntu:~/Documents/WBC/Commercial/DFA/complexWB-AES$
./complexWB-AES-mod testtesttesttest
FF1E4C03844DD800CE9CF34C6B7EEAE8 ← faulty cipher-text
```

DFA ON COMPLEX-CIPHER

Plaintext(in hex)

0x74657374746573747465737474657374

Correct ciphertext

0x6CB721A5633DFD7F94A6474524789026

Good faulty ciphertexts

0x8EB721A5633DFDB794A61E4524359026
0x6CD921A5113DFD7F94A647982478 6026
0x6CB740A563C5FD7F8BA647452478 90A8
0x6CB72161633DA97F94A44745 3D789026

… other 625 good faulty ciphertext

Final round key 0x5CB2FAF4F3FB94543BFA87DFE92660FC

DFA ON COMPLEX-CIPHER

Final round key	0x5CB2FAF4F3FB94543BFA87DFE92660FC
Round 9 key	0xEC26DC41AF496EA0C801138BD2DCE723
Round 8 key	0x36991EE3436FB2E167487D2B1ADDF4A8
Round 7 key	0x9C3EF21C75F6AC022427CFCA7D958983
Round 6 key	0xEB64C9D7E9C85E1E51D163C859B24649
Round 5 key	0x305BC5E702AC97C9B8193DD608632581
Round 4 key	0xFAF69E0032F7522EBAB5AA1FB07A1857
Round 3 key	0x78C1CC67C801CC2E8842F8310ACFB248
Round 2 key	0x21177A74B0C000494043341F828D4A79
Round 1 key	0xA8E4495191D77A3DF0833456C2CE7E66
Secret key	0x4A324D723933336C61544E6B324D4A30

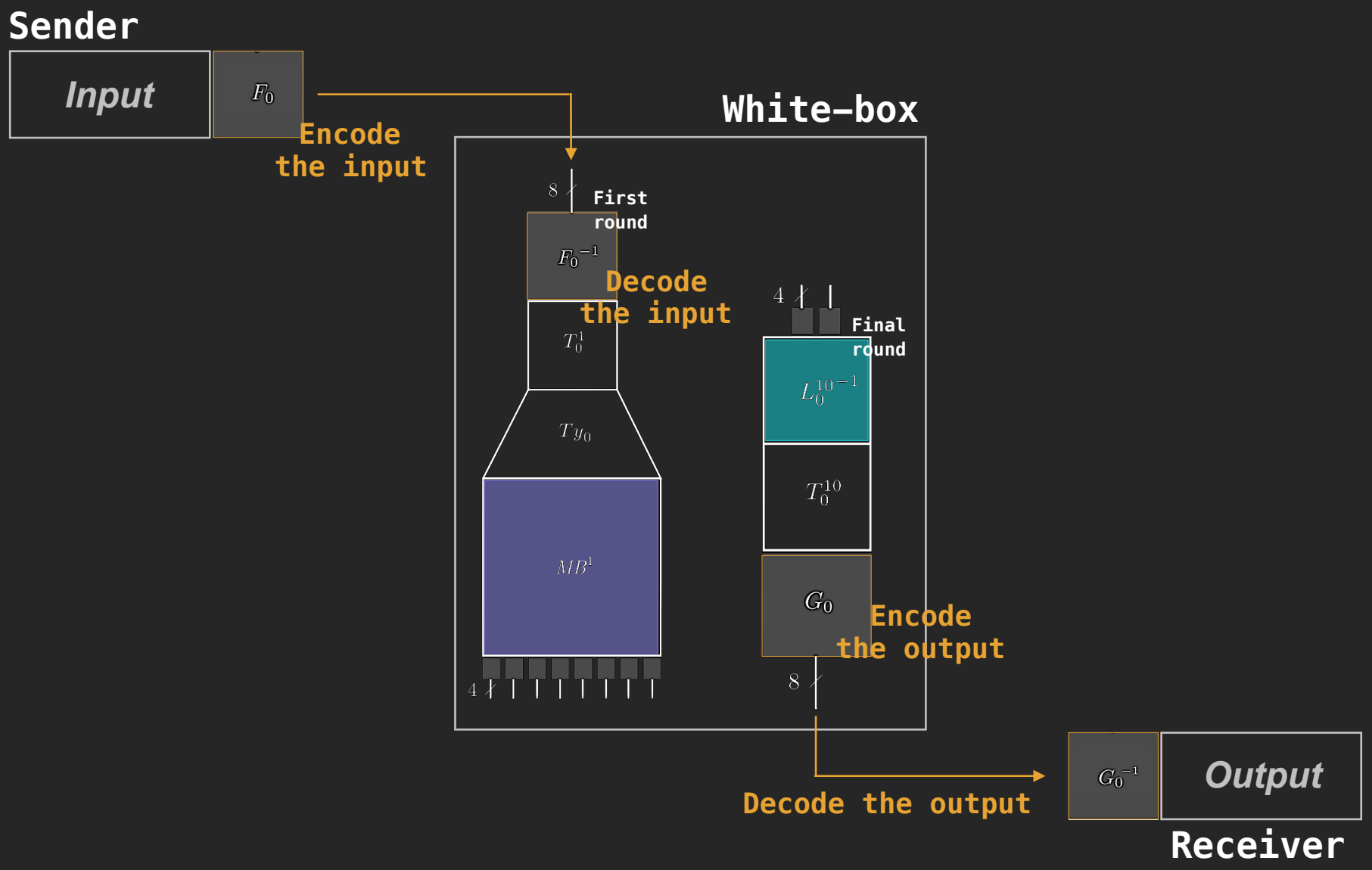
Conclusion

The way to use WBC safer in apps

Be sure to keep basic rules

- No single key for everything
- No hardcoded key(protected key)
- No static IV
- External encoding
- Asymmetric crypto algorithm based on WBC
  - RSA, Elliptic curves, Diffie–Hellman
- Tamper resistant embedded integrity checksums
- Cryptographic key device binding
  - device identifier + user identifier + external identifier(e,g, pin, biometric)

EXTERNAL ENCODING



Cryptographic Key Device binding

- Device identifier
- User identifier
- External identifier(e,g, pin, biometric)

Future works

- White-box version of crypto libraries
- Retrieve a master key embedded white-box engine
- Since the vulnerabilities have already been fixed,
  I will focus on finding other vulnerability.

THANK YOU