

Universal Immobilizer Crypto Engine

“U I C E, the little brother of AES”

AES4 Conference, 10.May.2004, Bonn, Germany

Dr. Ulrich Kaiser

Texas Instruments

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 1

Content

- Motivation
- Immobilizer
- UICE Concept
- Conclusion
- Appendix

Literature
Different S-Boxes
Pseudo-Code of Sensitivity Test

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 2

Motivation (1)

- Different Proprietary Cryptographic Algorithms in Immobilizers are in use today
- Fear to disclose algorithm details because weakness may be discovered
- Fear of vehicle call back in large numbers
- Information Hiding → Security ?
- Is a 'proof' for minimum hardness of the algorithm available ?

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 3

Motivation (2)

- OEM has to trust the supplier that he 'did it right' (know-how problem)
- OEM is bound to one supplier for a certain model, has no second source
- Use Challenge-Response Protocol
- → Create an Universal Crypto Engine for Immobilizers and make this an Industry Standard.

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 4

Content

- Motivation
- Immobilizer
- UICE Concept
- Conclusion
- Appendix

Literature
Different S-Boxes
Pseudo-Code of Sensitivity Test

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 5

Immobilizer System

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 6

Authentication

- Service to guarantee the correct entity and/or the correct information
- Not identical with secrecy !!!
- Real-time or delayed message traffic
- Challenge-Response protocol
- Unilateral authentication – mutual authentication

Identification

- Service for entity authentication
- Both entities are present now (real-time)
- If Alice successfully authenticates herself to Bob, then Bob will accept Alice's identity
- Bob cannot reuse the exchange with Alice to successfully impersonate Alice
- It is infeasible for Carol to play the role of Alice, causing Bob to accept Carol as Alice
- The features above remain true, even if an enemy observes large numbers of honest exchanges between Alice and Bob

Challenge-Response

- The verifier sets a challenge to the claimant, and only the true claimant can give the correct response.
- Challenges chosen at **random** !!
- State-of-the-Art Method for Authentication

Challenge-Response

Random Number Generation (RNG)

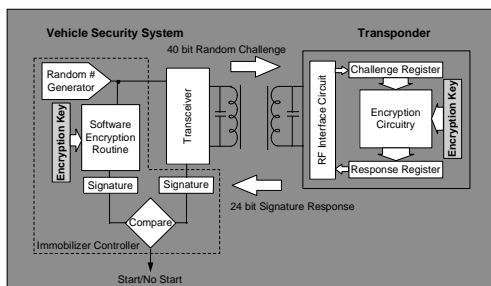


Strong Encryption Algorithm
+
Weak Random Number Generator
→
Bad System

(see NIST standards for RNGs)

Could you jump again ?

Immobilizer Transponder



Only 24 bit Response in order to make Dictionary Attack hard

Challenge-Response Principles

- Before any part of the response is transmitted all challenge must be processed and all key bits must be fully used
- Otherwise the first part of the transmission will not depend on part of the challenge or on part of the key

Well designed Block Cipher

- Every ciphertext bit is function of every plaintext bit and every key bit
- Avalanche effect causes massive error propagation
- All processing before transmission

Iterated Block Cipher

- Simple function repeated many times
- Complexity comes from the repetition
– more than exponential

Content

- Motivation
- Immobilizer
- UICE Concept
- Conclusion
- Appendix

Literature
Different S-Boxes
Pseudo-Code of Sensitivity Test

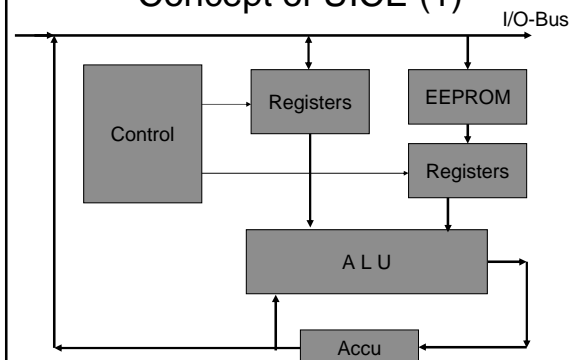
Universal Immobilizer Crypto Engine UICE Requirements (1)

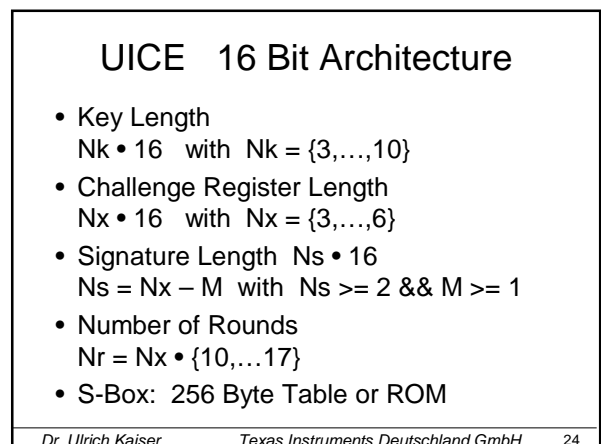
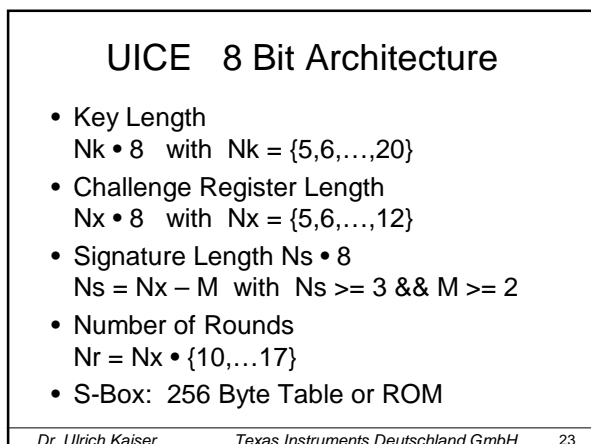
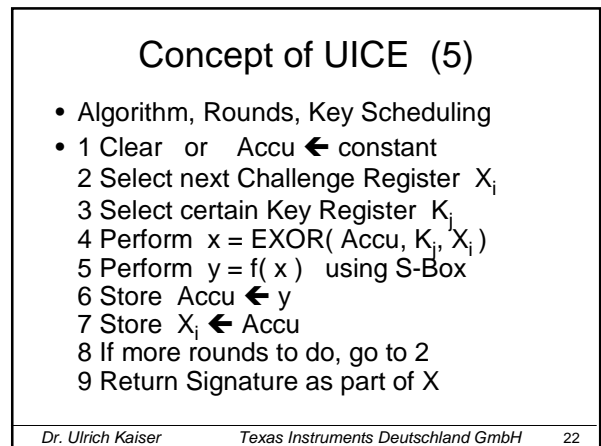
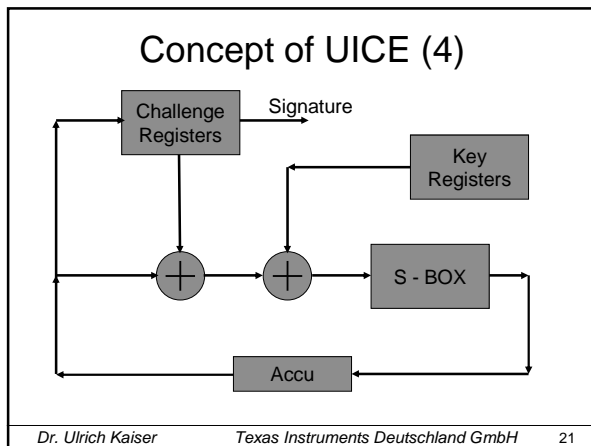
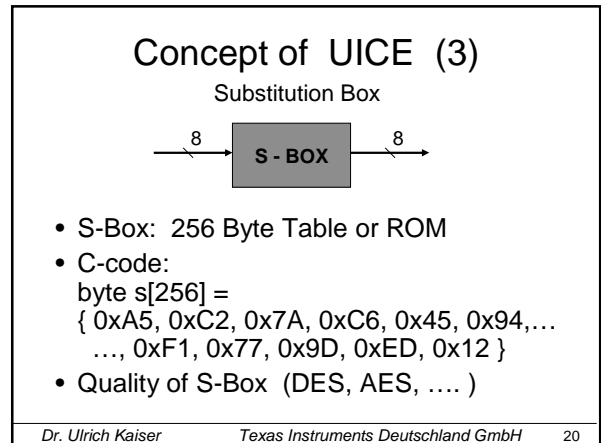
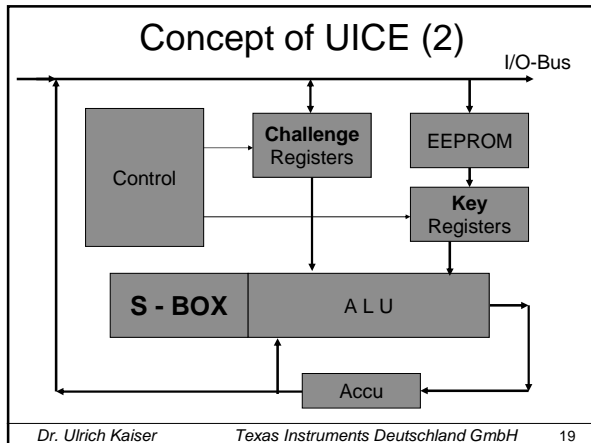
- Scalable Key Length
- Scalable Challenge Register Length
- Scalable Signature Length
- Scalable Number of Rounds
- Efficient Implementation in μC Software
- Efficient Implementation in Hardware
- Architecture Option: 8 bit / 16 bit (Mode)

UICE Requirements (2)

- Not 'Obscurity by Complexity'
- Strong Key Avalanche Effect
- Protection against Cryptanalysis
 - Differential Cryptanalysis
 - Linear Cryptanalysis
 - Dictionary Attacks
 - Related Key Cryptanalysis
 - Timing Attacks
 - Power Analysis Attacks

Concept of UICE (1)





UICE Details (1)

- Learn from AES
- Run tests :
 - Sensitivity (challenge, key) → Avalanche
 - Correlation (challenge, key)
 - Affinity (challenge, key) → Linearity
 - RNG (FIPS 140-2)
 - other
- Measure speed

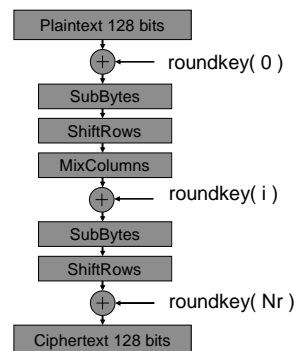
AES (1)

- Advance Encryption Standard
- NIST Proposal, Jan.1997
- Goal: Efficient in Hardware AND Software
- Preliminary candidates: 15
- Final candidates: 5 (MARS, RC6, Rijndael, Serpent, Twofish)
- Final selection Oct.2000:
 - Rijndael** (block size = 128)
 - J.Daemen, R.Rijmen, KU Leuven

AES (2)

- block size: 128 bits
- Architecture: Iterated Block Cipher with operations on a Two-Dimensional State Array
- key size: 128, 192, or 256 bits
- Rounds: 10, 12, or 14 (Nr)
- f(): one S-Box
- Key schedule: substitutions using S-box, rotations of bytes in words, XOR with constants

AES (3)



AES (4)

- SubBytes: substitution, S-Box, built using multiplicative inverse of each element in $GF(2^8)$ with irreducible polynomial $m(x)=x^8+x^4+x^3+x+1$ and additional affine transformation
- ShiftRows: cyclically shift of bytes, dependent on row (e.g. 4th row → shift by 3 bytes)
- MixColumns: permutation using $GF(2^8)$ multiplication by $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \pmod{x^4 + 1}$

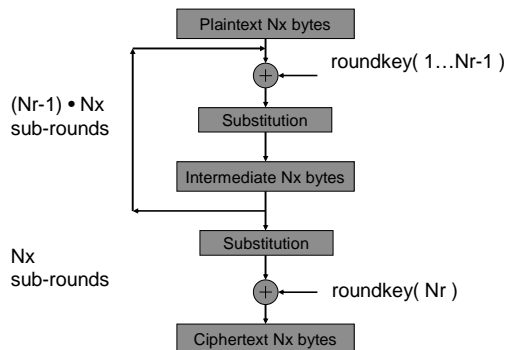
AES (5)

- Speed on Xilinx XCV1000 FPGA
 - 11k slices, 31.8 MHz clock, 1938 Mbit/s
 - 3k slices, 40.4 MHz clock, 492 Mbit/s
- Speed on Pentium Pro 200MHz
 - 27.0 Mbit/s (ANSI C)
 - 70.5 Mbit/s (Visual C++)
 - 1.1 Mbit/s (Java 1.1.1)
- Length on Intel 8051 (12 osc.period = 1 cycle)
 - 3168 cycles, 1016 bytes
 - 4065 cycles, 768 bytes
- Length on Motorola 68HC08 (1 osc.per. =1 cycle)
 - 8390 cycles, 919 bytes

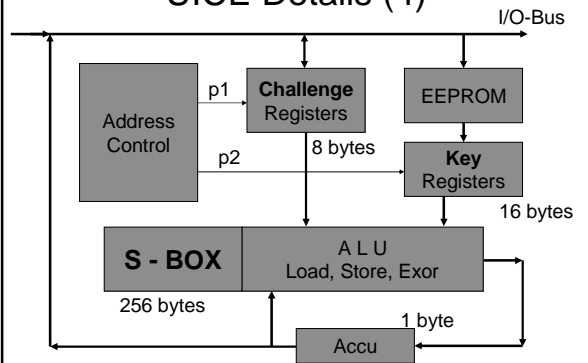
UICE Details (2)

- Learn from AES
 - Key applied at the beginning
 - Key applied at the end !
 - Use Rijndael S-Box
- Use also other random generated S-Boxes
- Keep key scheduling simple

UICE Details (3)



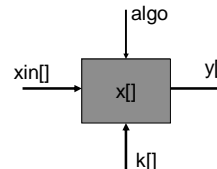
UICE Details (4)



UICE C-Code (1)

```
void crunch_UICE( unsigned char xin[], unsigned char y[],
                unsigned char k[], int algo )
{
    int round; /* round counter */
    int i; /* inner counter */
    int nx; /* number of challenge bytes */
    int nk; /* number of key bytes */
    int rounds; /* number of rounds to run */
    int p1; /* pointer to actual challenge byte */
    int p2; /* pointer to actual key byte */
    unsigned char accu;
    unsigned char x[16]; /* Local working register for challenge and result */

    x[0] = xin[0]; /* Copy needed in order to */
    x[1] = xin[1]; /* avoid overwriting xin ! */
    x[2] = xin[2]; /* This is not needed in hardware. */
    x[3] = xin[3];
    x[4] = xin[4];
    nx = 5;
    nk = 5;
    if( algo == 4 || algo == 5 ) /* 8 bytes challenge */
    {
        x[5] = xin[5];
        x[6] = xin[6];
        x[7] = xin[7];
        nx = 8;
        nk = 8;
        if( algo == 5 ) nk = 16;
    }
}
```



UICE C-Code (2)

```
#define SBOXx sbboxAF
#define SBOXy sbbox130
#define SBOX S
#define NR 10
/* ----- start of algorithm ----- */
nrounds = NR; /* number of rounds, i.e. outer loop count ! */
accu = 0;
p1 = 0;
p2 = 0;
for( round=1; round <= nrounds-1; round++ )
{
    for( j=1; j <= nx; j++ )
    {
        accu = accu ^ x[p1] ^ k[p2]; /* linear operation */
        accu = SBOX[ accu ]; /* key mixing before sbox operation */
        accu = SBOX[ accu ]; /* NON-LINEAR OPERATION */
        x[p1] = accu; /* overwrite challenge byte ! */
        /* update the pointers */
        p1++;
        if( p1 >= nx ) p1 = 0; /* 0..4 or 0..7 */
        p2 += 3;
        if( p2 >= nk ) p2 = p2 - nk;
    } /* for j */
    /* key scheduling so that x[] sees different k[] */
    if( round==2 ) p2++;
    else if( round==4 ) p2++;
    else if( round==6 ) p2++;
    else if( round==8 ) p2++;
    if( p2 >= nk ) p2 = p2 - nk;
} /* for round */
```

UICE C-Code (3)

```
/* final round */
for( j=1; j <= nx; j++ )
{
    accu = accu ^ x[p1]; /* linear operation */
    accu = SBOX[ accu ]; /* NON-LINEAR OPERATION */
    accu = accu ^ k[p2]; /* key mixing after sbox operation */
    x[p1] = accu; /* overwrite challenge byte ! */
    /* update the pointers */
    p1++;
    if( p1 >= nx ) p1 = 0; /* 0..4 or 0..7 */
    p2 += 3;
    if( p2 >= nk ) p2 = p2 - nk;
} /* for j */
/* ----- end of algorithm ----- */
y[0]=x[0]; /* Copy result to output; this is not needed in hardware ! */
y[1]=x[1];
y[2]=x[2];
y[3]=x[3];
y[4]=x[4];
if( algo == 4 || algo == 5 )
{
    y[5] = x[5];
    y[6] = x[6];
    y[7] = x[7];
}
} /* crunch_UICE ----- */
```

Run Times of Different Cryptographic Algorithms

| Algorithm | time us | Challenge Bytes | Key Bytes | Cipher rounds | Notes |
|-----------|---------|-----------------|-----------|---------------|-----------|
| IMMO | 480 | 5 | 5 | 10 | {1} |
| UICE40 | 5.6 | 5 | 5 | 10 | {2} |
| UICE64 | 8.6 | 8 | 8 | 10 | {2} |
| UICE128 | 8.6 | 8 | 16 | 10 | {2} |
| AES128 | 163 | 8 | 16 | 10 | {3,4}[14] |

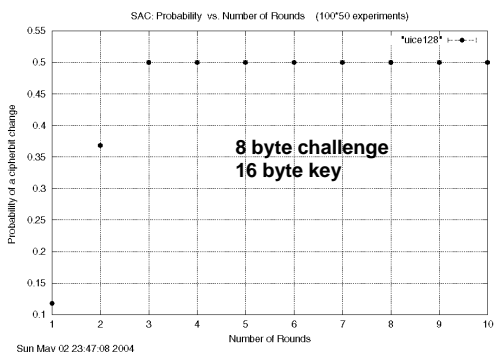
Notes:
 (1) IMMO is based on many shift register operations, slow in software.
 (2) UICE is based on Byte-Operations, is fast in software.
 (3) AES (i.e. Rijndael) was chosen as proper compromise between hardware and software applications.
 (4) AES128 is based on rijndael-alg.ref.c and needs three tables of 256 bytes each and also a table of 30 bytes [pulled from <http://www.esat.kuleuven.ac.be/~rijmen/rijndael/index.html>]. However, the code for key length 192 und 256 was eliminated. The sixteen bytes per block are filled with eight bytes of the Challenge, the rest is filled with zeroes. As result only eight bytes are taken.

UICE Quality Tests

- Sensitivity (challenge, key) → Avalanche
- Correlation (challenge, key)
- Affinity (challenge, key) → Linearity
- RNG (FIPS 140-2)
- Byte Linearity Test
- Byte Differential Test

All these tests **passed** for three different S-Boxes and the three algorithms.

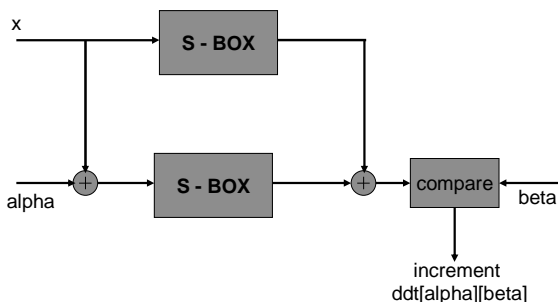
Strict Avalanche Criterion (Key)



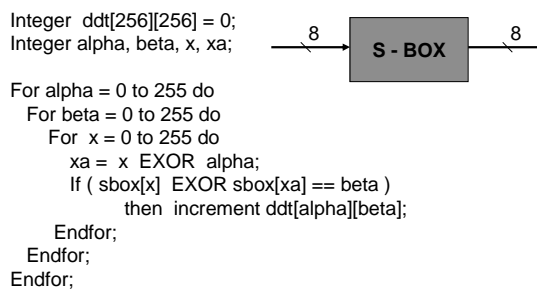
Key Sensitivity Test



Differential Distribution Table (1)



Differential Distribution Table (2)



Differential Distribution Table

Results for Different S-Boxes

| Name | CodeName | 2 | 4 | 6 | 8 | 10 | 12 |
|--------------|----------|-------|------------|----------|----------|-----------|----------|
| Rijndael | S | 32130 | 255 | 0 | 0 | 0 | 0 |
| Rijndael-Inv | | 32130 | 255 | 0 | 0 | 0 | 0 |
| random 1 | sboxAF | 19763 | 4917 | 854 | 106 | 9 | 2 |
| random 2 | | 19661 | 4954 | 855 | 109 | 14 | 0 |
| random 3 | sboxL30 | 22166 | 4629 | 400 | 4 | 0 | 0 |
| (random-goal | | 28000 | 4000 | 0 | 0 | 0 | 0) |

Comparison of Different Algorithms

| Property / Name | UICE128 | IMMO | AES |
|-----------------|-------------|-------------|-----------------|
| Complexity | low | high | medium |
| Speed Software | high | low | medium |
| Speed Hardware | high | medium | medium |
| Chip Area | low | medium | medium |
| Security | higher | high | highest |
| Usage | signature | signature | mass data |
| Configuration | ECB | ECB | ECB, CBC, etc. |
| Application | immobilizer | immobilizer | data encryption |

Content

- Motivation
- Immobilizer
- UICE Concept
- Conclusion
- Appendix

Literature
Different S-Boxes
Pseudo-Code of Sensitivity Test

Conclusions (1)

- A new scalable Cryptographic Engine for Immobilizers has been proposed.
- The architecture has moderate complexity and is suited for efficient hardware and software implementations.
- The algorithm is strong regarding typical attack methods
- Optional 8 bit or 16 bit word length.
- Requirements for the defense of modern attack methods are considered.

Conclusions (2)

- Promotion at major customers
- Make it an Open Industry Standard !!!
- Need evaluation by some independent institutes, e.g. CCI (Meppen), T-Systems (Bonn),...

End of UICE

- Thank you for your attention !
- My email address: d-kaiser@ti.com

Appendix follows....

Content

- Motivation
- UICE Concept
- Conclusion
- Appendix
 - Literature
 - Different S-Boxes
 - Pseudo-Code of Sensitivity Test

Literature (1)

- [1] John Gordon, Introduction to Cryptography, ConceptLabs, 1998
- [2] A. Menezes, P. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997
- [3] D. Stinson, Cryptography - Theory and Practice, CRC Press, 1995
- [4] B. Schneier, Applied Cryptography, Wiley, 1994
- [5] R. Anderson, Why Cryptosystems fail, 1st Conference on Computer and Communication Security '93, ACM, Nov. 1993
- [6] J. Daemen, V. Rijmen, AES Proposal: Rijndael, Version 2, 03/09/99, 45 pages and related Reference Code in C <http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaelref.zip>
- [7] J. Seberry, X. Zhang, Y. Zheng, Pitfalls in Designing Substitution Boxes, S-Box, Crypto'94, Aug.1994, pp 383ff
- [8] T. Wollinger, M. Wang, J. Guajardo, C. Paar, How Well are High-End DSPs Suited for the AES Algorithms ? - AES Algorithms on the TMS320C6x DSP, The Third Advance Encryption Standard (AES3) Candidate Conference, April 2000, New York, 11 pages

Literature (2)

- [9] M. Loney, Moore's Law is the biggest threat to privacy, according to Phil Zimmermann, news.zdnet.co.uk and www.silicon.com, 29.Apr.2003
- [10] NIST FIPS 140-2, Security Requirements for Cryptographic Modules, May 25, 2001, <http://csrc.nist.gov/cryptval> and <http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf>
- [11] J. Gordon, A. Retkin, Are Big S-Boxes Best ?, IEEE Workshop on Communication Security, Santa Barbara, Cal. 1981, pp. 1-6
- [12] H. Heys, S. Tavares, Substitution-Permutation Network Resistant to Differential and Linear Cryptanalysis, Journal of Cryptology, Vol. 9, No. 1, pp.1-19, 1996
- [13] J. Rejeb, V. Ramaswamy, K. Ghadiri, Hardware Implementation of the Rijndael Algorithm for High-Speed Networks, ISPC 2003, March 2003, Dallas, 6 pages
- [14] H. Kuo, I. Verbauwhede, Architectural Optimization for a 1.82Gbits/sec VLSI Implementation of the AES Rijndael Algorithm, CHES 2001, LNCS 2162, pp. 51-64, Springer 2001

Literature (3)

- [15] National Institute of Standards and Technology, FIPS PUB 140-2 Annex A: Approved Security Functions, www.nist.gov/cmvp.
- [16] Kocher, Jaffe, Jun, Differential Power Analysis, Advances in Cryptology, CRYPTO'99, LNCS 1666, Springer 1999, 10 pages
- [17] Kocher, Evaluating Cryptosystems, 31 slides, Cryptography Research 2002, <http://www.cryptography.com/resources/whitepapers/HackingCryptosystems.pdf>

AES S-Box beginning with 99

```
word8 S[256] = {
99, 124, 119, 123, 242, 107, 111, 197, 48, 1, 103, 43, 254, 215, 171, 118,
202, 130, 201, 125, 250, 89, 71, 240, 173, 212, 162, 175, 156, 164, 114, 192,
183, 253, 147, 38, 54, 63, 247, 204, 52, 165, 229, 241, 113, 216, 49, 21,
4, 199, 35, 195, 24, 150, 5, 154, 7, 18, 128, 226, 235, 39, 178, 117,
9, 131, 44, 26, 27, 110, 90, 160, 82, 59, 214, 179, 41, 227, 47, 132,
83, 209, 0, 237, 32, 252, 177, 91, 106, 203, 190, 57, 74, 76, 88, 207,
208, 239, 170, 251, 67, 77, 51, 133, 69, 249, 2, 127, 80, 60, 159, 168,
81, 163, 64, 143, 146, 157, 56, 245, 188, 182, 218, 33, 16, 255, 243, 210,
205, 12, 19, 236, 95, 151, 68, 23, 196, 167, 126, 61, 100, 93, 25, 115,
96, 129, 79, 220, 34, 42, 144, 136, 70, 238, 184, 20, 222, 94, 11, 219,
224, 50, 58, 10, 73, 6, 36, 92, 194, 211, 172, 98, 145, 149, 228, 121,
231, 200, 55, 109, 141, 213, 78, 169, 108, 86, 244, 234, 101, 122, 174, 8,
186, 120, 37, 46, 28, 166, 180, 198, 232, 221, 116, 31, 75, 189, 139, 138,
112, 62, 181, 102, 72, 3, 246, 14, 97, 53, 87, 185, 134, 193, 29, 158,
225, 248, 152, 17, 105, 217, 142, 148, 155, 30, 135, 233, 206, 85, 40, 223,
140, 161, 137, 13, 191, 230, 66, 104, 65, 153, 45, 15, 176, 84, 187, 22,
};/* ddt: 32130 255 0 0 0 0 */
```

Random S-Box beginning with 0xAF

```
unsigned char sboxAF[256] =
{0xAF, 0x1B, 0xDD, 0xBC, 0x30, 0xEB, 0xF0, 0x56, 0xC1, 0x08, 0x93, 0x36, 0x03, 0xCB,
0x81, 0x80, 0x43, 0x2F, 0xDF, 0x2D, 0x26, 0x05, 0x0A, 0xF8, 0x7D, 0x21, 0xE0, 0xC4,
0x06, 0xD5, 0xA6, 0xE8, 0x8E, 0x70, 0xDC, 0xA4, 0x6D, 0x23, 0xAC, 0x18, 0x40, 0x00,
0x64, 0x0E, 0xF6, 0x79, 0xB5, 0x1F, 0x5D, 0x9A, 0x3B, 0xFA, 0x48, 0x5F, 0x74, 0xA1,
0x8D, 0xD3, 0x5C, 0x4E, 0x9E, 0x14, 0x25, 0xEF, 0xD6, 0xC9, 0x3F, 0xC5, 0xA0, 0x10,
0x50, 0xFB, 0x31, 0xF4, 0x17, 0xB8, 0xAB, 0x32, 0x76, 0x3E, 0x15, 0x2A, 0x3D, 0xA9,
0x52, 0x20, 0xC3, 0xFC, 0x7B, 0x49, 0x3A, 0x6E, 0xB7, 0x1D, 0xAD, 0xAA, 0x5A, 0x0D,
0x35, 0x38, 0xC8, 0xF5, 0xF3, 0xB3, 0x8F, 0xE6, 0x13, 0x55, 0x33, 0x8A, 0xC0, 0x67,
0x2E, 0xE7, 0xB2, 0x8C, 0x09, 0xCF, 0x1E, 0x97, 0x28, 0x07, 0xBA, 0x4D, 0x42, 0x04,
0x73, 0x41, 0x5B, 0xB1, 0xF9, 0xE5, 0xFF, 0x6C, 0xD8, 0x12, 0xB5, 0x84, 0xCC, 0xB0,
0x69, 0x37, 0xAE, 0x6F, 0xE2, 0xDB, 0x0C, 0x86, 0x29, 0x78, 0x34, 0x7C, 0x1A, 0x85,
0x27, 0xA3, 0x9B, 0x92, 0xE3, 0xBD, 0x59, 0x63, 0x66, 0x19, 0xCA, 0x6E, 0xFF, 0x75,
0x72, 0x24, 0x4F, 0x47, 0x61, 0x11, 0x0B, 0xBE, 0xA7, 0x16, 0x3C, 0xB2, 0xFD, 0x7F,
0x44, 0x99, 0x6B, 0x38, 0x22, 0x46, 0x4A, 0x1C, 0x02, 0x6A, 0x51, 0x39, 0x60, 0x4E,
0x57, 0x01, 0x2C, 0xE1, 0xEE, 0x83, 0x89, 0xDA, 0x58, 0x0F, 0xB8, 0x2B, 0xD2, 0xD4,
0x62, 0x9F, 0x90, 0x7E, 0xDE, 0xB8, 0x4C, 0xCD, 0x68, 0xA8, 0xF2, 0x54, 0xE9, 0x4,
0xF1, 0xEA, 0xD7, 0x77, 0x9D, 0x96, 0xEC, 0xFE, 0xB9, 0x91, 0xBF, 0xD1, 0xD9, 0xA2,
0x95, 0xED, 0xA5, 0xC2, 0x7A, 0xC6, 0xC7, 0x45, 0x94, 0xB6, 0x65, 0xD0, 0xCE, 0x9C,
0x71, 0x87, 0xB4, 0x53};/* ddt: 19763 4917 854 106 9 2 0 */
```

Random S-Box beginning with 130

```
unsigned char sbox130[256] =
{130,150,219,161,127,160,229,198,99,26,22,63,74,136,215,201
,82,195,3,225,239,94,129,80,18,213,149,245,7,57,197,115
,113,230,116,163,212,133,162,222,105,60,19,170,244,30,1,137
,176,27,185,42,153,16,104,202,221,11,172,190,154,151,103,71
,28,187,2,88,231,204,17,37,228,73,44,31,134,100,144,211
,89,117,108,39,227,241,232,247,20,226,110,254,169,14,174,119
,131,152,55,205,49,164,142,43,132,12,98,224,135,157,114,83
,78,236,111,23,147,70,47,252,189,32,41,124,120,58,102,33
,234,184,158,177,140,121,246,180,175,45,101,233,168,138,203,188
,15,97,183,196,59,250,54,6,148,207,72,118,206,86,48,112
,199,36,96,145,75,85,220,186,106,217,34,240,87,178,69,65
,238,91,179,159,249,146,107,214,56,141,10,243,125,165,8,29
,52,13,253,193,66,21,126,50,77,251,143,84,192,25,9,79
,93,46,81,0,38,4,35,5,194,95,128,242,122,156,109,90
,248,166,61,76,167,67,210,155,139,255,24,40,53,62,216,218
,173,181,200,68,191,223,171,64,209,92,123,237,182,235,51,208
}; /* dt: 22166 4629 400 4 0 0 0 */
```

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 55

Tries to find sensitivities between input bit and output bit,
i.e. check if avalanche effect is sufficient,
a) between key bit and response bit (described below)
b) between challenge bit and response bit

```
Details:
using: crunch( challenge, response, key );
n = 50;
Do for 100 experiments
Do for n runs
/* setup random challenge and random key */
for( j=0; j<CHALLENGEBYTES; j++) x[j] = RandByte();
for( j=0; j<KEYBYTES; j++) k[j] = RandByte();
/* find yy[] as a reference */
crunch( x, yy, k );
for all KEYBITS do
toggle k [ KEYBIT ];
crunch( x, y, k );
/* y and yy should be different ! */
for all RESPONSEBITS do
if( y[ RESPONSEBIT ] == yy[ RESPONSEBIT ] )
then increment hist[ RESPONSEBIT ][ KEYBIT ];
end
toggle k [ KEYBIT ]; //restore k
end
end
Find HIGHS and LOWs in histograms
Print final results.
Find HIGHS and LOWs in histograms
```

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 56

```
#define CHISQ_90 0.0157908
#define CHISQ_10 2.70554
CHISQ_LO = CHISQ_90;
CHISQ_HI = CHISQ_10;
do for all key bits // CHI**2 Test
do for all response bits
chisq = ( 2 * hist[j] - n );
chisq = chisq * chisq / n;
if( chisq > CHISQ_HI ) highs[j]++;
if( chisq < CHISQ_LO ) lows[j]++;
hist[j] = 0;
end
end
Print final results
-----
print for every response bit the HIGHS / number of experiments in %;
print for every response bit the LOWs / number of experiments in %.
Horizontal: response bit position
Vertical : key bit position
print histograms
Expected result: about 10% for every bit position
```

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 57

Disclaimer

- Neither the author nor any organization associated with him will be responsible for any consequences or losses resulting directly or indirectly from material contained in this document or for any errors or omissions.
- Some suggestions contained here may be subject of patent applications of third parties. Readers are warned to check on possible intellectual property rights before implementing or selling systems using information or suggestions found here.
- The export of cryptographic devices is controlled in many countries. Manufacturers are urged to consult relevant authorities.

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 58

The End

- Thank you for your attention !
- My email address: d-kaiser@ti.com

Dr. Ulrich Kaiser Texas Instruments Deutschland GmbH 59