**SCIENCE CHINA**
Information Sciences

• **Supplementary File** •

# Fast Substitution-Box Evaluation Algorithm and Its Efficient Higher-Order Masking Scheme for Block Ciphers

Hai Huang[1,2], Leibo Liu[1*], Min Zhu[1], Shouyi Yin[1] & Shaojun Wei[1]

[1]*Institute of Microelectronics, Tsinghua University, Beijing* 100084*, China;*
[2]*Harbin University of Science and Technology, Harbin* 150080*, China*

## Appendix A Proof of Theorem 1

If $a$ is the primitive root, then all elements over $GF(2^n)$ can be generated from (A1).

$$x_i = a^i (mod\ p), i \in [0, 2^n - 1] \tag{A1}$$

Since,

$$x^{2^n-1} \equiv x^{mk} \equiv 1(mod\ p) \tag{A2}$$

Here, $a^k$ is one solution of $x^m \equiv 1(mod\ p)$, hence all solutions satisfy,

$$x_j = a^{j^k} (mod\ p), j \in \mathbb{N} \tag{A3}$$

Since the equation only has $m$ solutions, hence this theorem is proven.

## Appendix B Proof of Theorem 2

The $k$ of $m$-degree congruence equations can be constructed as:

$$\begin{cases} x^m \equiv b_1 \equiv 1(mod\ p) \\ x^m \equiv b_2(mod\ p) \\ \cdots\cdots\cdots \\ x^m \equiv b_{k-1}(mod\ p) \\ x^m \equiv b_k(mod\ p) \end{cases} \tag{B1}$$

Since $b_j, j \in [2, k]$ are solutions of $x^k \equiv 1(mod\ p)$, then

$$b_j = a^{(j-1)^m} (mod\ p), j \in [1, m] \tag{B2}$$

Hence, multiplying both sides of (B1) by $b_j$ from (B2) proves this theorem.

* Corresponding author (email: liulb@tsinghua.edu.cn)

## Appendix C Table C1

**Table C1** Solutions of the high-degree congruence with factoring pairs

| $n$ | Factors | $p$ | $m$ | $x$ satisfies $x^m \equiv 1(mod\ p)$ |
|---|---|---|---|---|
| 4 | (3, 5) | $x^4+x+1$ | 3 | 1, 6, 7 |
| | | | 5 | 1, 8, 10, 12, 15 |
| 6 | (3, 21) | $x^6+x+1$ | 3 | 1, 58, 59 |
| | | | 21 | 1, 3, 5, 8, 13, 14, 15, 17, 18, 22, 23, 24, 25, 40, 43, 51, 54, 57, 58, 59, 62 |
| | (7, 9) | | 7 | 1, 14, 15, 22, 23, 24, 25 |
| | | | 9 | 1, 6, 11, 20, 26, 28, 31, 58, 59 |
| 8 | (3, 85) | $x^8+x^4+x^3+x^1+1$ | 3 | 1, 188, 189 |
| | | | 85 | 1, 7, 8, 10, 12, 15, 21, 22, 29, 36, 38, 41, 43, 45, 46, 47, 50, 52, 53, 54, 55, 56, 57, 59, 61, 64, 66, 67, 68, 74, 80, 83, 85, 96, 98, 99, 102, 103, 107, 111, 115, 117, 120, 124, 127, 130, 133, 137, 139, 140, 146, 156, 158, 161, 162, 163, 168, 171, 174, 175, 176, 179, 181, 182, 187, 194, 195, 199, 202, 205, 209, 210, 211, 213, 217, 219, 223, 232, 237, 239, 242, 243, 247, 249, 252 |
| | (5, 51) | | 5 | 1, 12, 80, 176, 237 |
| | | | 51 | 1, 2, 4, 8, 16, 27, 29, 32, 37, 47, 51, 53, 54, 57, 58, 64, 74, 77, 94, 97, 99, 102, 106, 108, 114, 116, 125, 128, 131, 141, 145, 148, 151, 154, 159, 171, 179, 188, 189, 194, 197, 198, 203, 204, 211, 212, 216, 228, 232, 239, 250 |
| | (15, 17) | | 15 | 1, 12, 13, 80, 81, 92, 93, 176, 177, 188, 189, 224, 225, 236, 237 |
| | | | 17 | 1, 8, 29, 47, 53, 54, 57, 64, 74, 99, 102, 171, 179, 194, 211, 232, 239 |
| 9 | (7, 73) | $x^9+x^4+1$ | 7 | 1, 28, 29, 332, 333, 336, 337 |
| | | | 73 | 1, 9, 23, 24, 41, 42, 43, 49, 60, 61, 62, 65, 70, 76, 88, 89, 94, 99, 102, 103, 111, 112, 127, 128, 137, 147, 156, 158, 162, 175, 183, 190, 191, 208, 216, 227, 255, 262, 266, 277, 286, 291, 298, 301, 308, 320, 326, 327, 331, 334, 353, 357, 362, 363, 364, 370, 371, 374, 378, 400, 406, 426, 441, 456, 457, 462, 469, 476, 481, 501, 502, 505, 511 |

## Appendix D Algorithm D1

**Algorithm D1** Higher-order masking scheme for GLUT

**Input:** Boolean shares $xa_0$, $xa_1$, $xa_2$, ..., $xa_d$ of $x$
**Output:** Boolean shares $y = GLUT(x) \oplus xb_1 \oplus xb_2 \cdots \oplus xb_d$, $xb_1$, $xb_2$, ..., $xb_d$
  **$BM$ to $MM$ operation
$(xm_0, xm_1, \cdots, xm_d) = BMtoMM(xa_0, xa_1, \cdots, xa_d)$; \\ Alg. D2
  ** *Table re-computation*
**for** $n = 1$ to $d$
Random generates $xb_n \in GF(2^n)$; \\ Masks refreshing
$GLUT'(u) \leftarrow GLUT(u \otimes xm_n) \oplus xb_n$;
  \\ Re-computation
$y \leftarrow GLUT'(xm_0)$;
**end**
**return** $y$, $xb_1$, $xb_2$, ..., $xb_d$.

## Algorithm D2

**Algorithm D2** BM to MM

**Input:** Boolean shares $xa_0$, $xa_1$, $xa_2$, ..., $xa_d$
**Output:** Multiplicative shares $xm_0$, $xm_1$, $xm_2$, ..., $xm_d$

```
xm_0 = xa_0;
for n = 1 to d
  Random generates i ∈ [1, k], j ∈ [1, m];
  xm_n = M_ij;
  xm_0 = xm_0 ⊗ xm_n;
  for k = 1 to d − n
    Random generates r ∈ [0, 255]; // Masks refreshing
    xa_k = xm_n ⊗ xa_k;
    xa_k = r ⊗ xa_k;
    xm_0 = r ⊕ xa_k;
    xa_k = r;
  end
  xa_{d−n+1} = xm_n ⊗ xa_{d−n+1};
  xm_0 = xm_0 ⊕ xa_{d−n+1};
end
return xm_0, xm_1, xm_2, ..., xm_d.
```

## Appendix E Table E1

**Table E1** Computing-complexity of different arithmetic modules.

| | XOR | Multiplication | LUT access |
|---|---|---|---|
| $F_2$-linear | 0 | 0 | $d+1$ |
| Multiplication over $GF(2^n)$ | $2(d^2+d)$ | $(d+1)^2$ | 0 |
| $x \times g(x)$ | $5(d^2+d)$ | 0 | $2d^2+3d+1$ |
| GLUT | $d^2+d$ | $(d(d+3))/2$ | $d$ |

## Table E2

**Table E2** Computing-complexity of different AEs S-boxes.

| | XOR | Multiplication | LUT access |
|---|---|---|---|
| [6] | $8(d^2+d)$ | $4(d+1)^2$ | $7(d+1)$ |
| [7] | $8(d^2+d)$ | $4(d+1)^2$ | $11(d+1)$ |
| [8] | $8(d^2+d)$ | $4(d+1)^2$ | $6(d+1)$ |
| [9] | $17(d^2+d)$ | $(d+1)^2$ | $6d^2+15d+10$ |
| This paper | $3(d^2+d)$ | $(3d^2+7d+2)/2$ | $5d+4$ |