IDEA BLOCK CIPHER FINAL PRESENTATION

Oleg Vyshnyvetskyi
Sébastien Guilloux

Team Osix

RIT – Cryptography course 2012

Summary

- What is IDEA ?
- Description of the algorithm
- Design of the software
  - IDEA
  - CBCMode
  - Usage
- Running time measurements
- Revised design
- What we learned

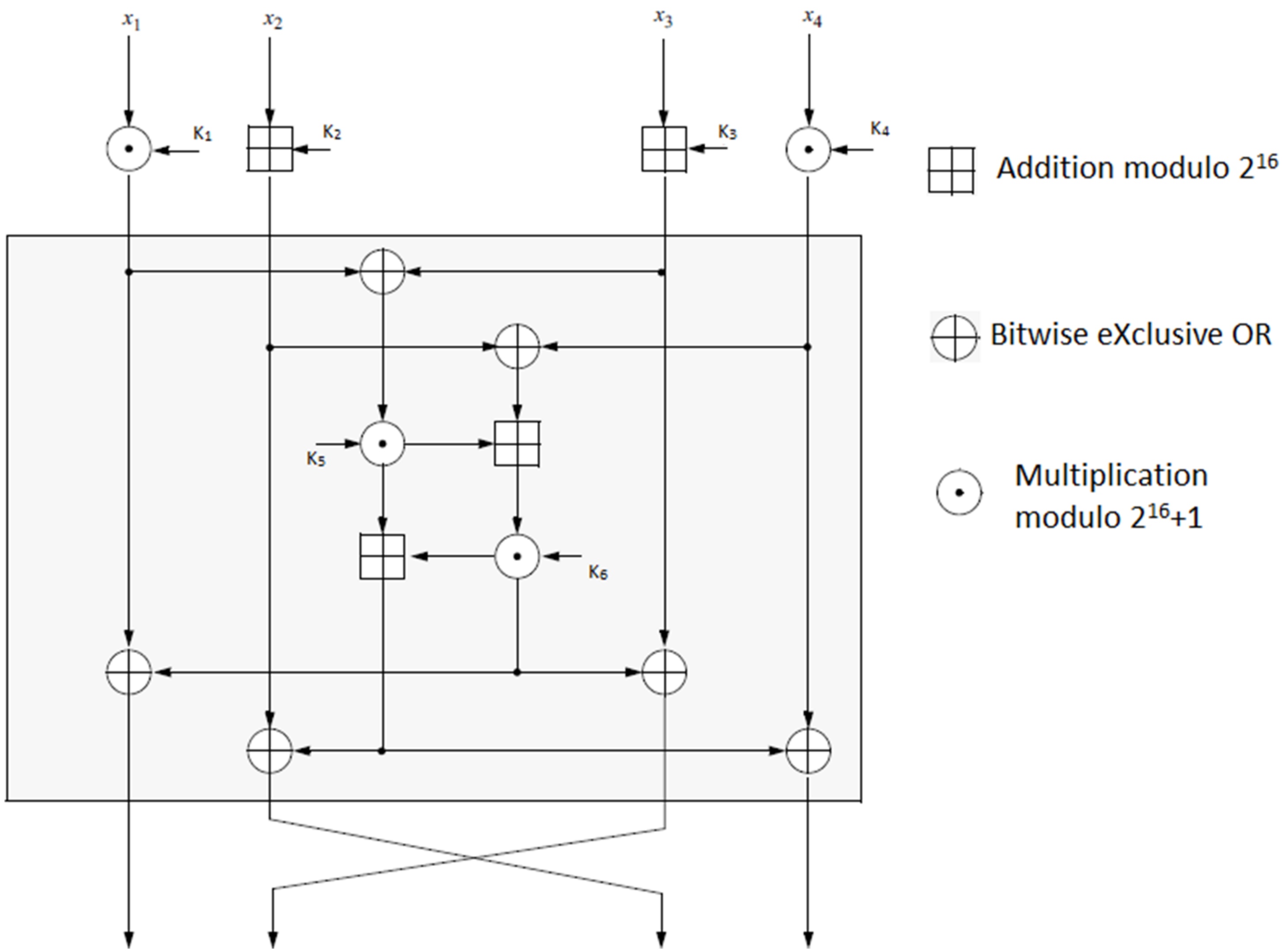
What is IDEA ?

- IDEA = International Data Encryption Algorithm
- Block Cipher
- Designed in 1991 by:
  - James Massey (cryptographer, Germany)
  - Xuejia Lai (cryptographer, China)

Algorithm - Overview

- Block cipher :
  - Block size : 64 bits
  - Key size : 128 bits
- 8 rounds + output transformation (half-round)
- Symmetric cipher

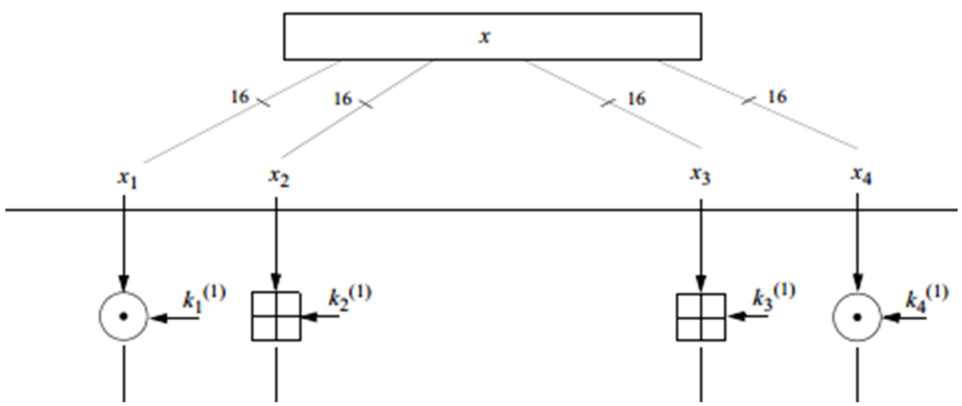
Round Structure (1)



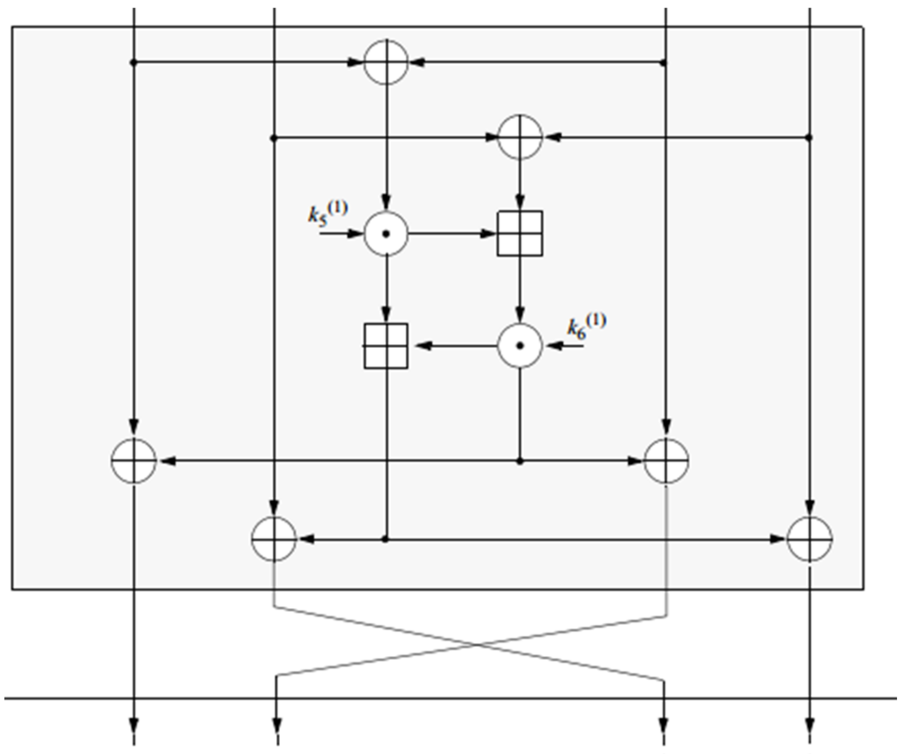
Round Structure (2)

- Split into four 16 bits quarter blocks
- **First Layer:**

  Apply two 16-bit additions and two 16-bit multiplications to quarter blocks using appropriate parts of the round key



Round Structure (3)

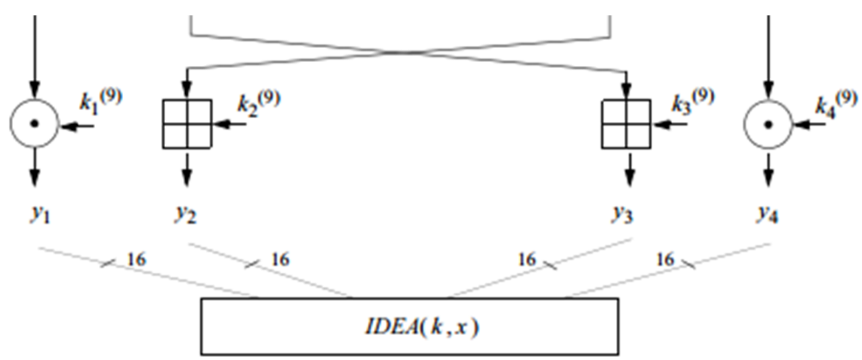


Second Layer

- Calculate two intermediate quarter blocks with 16-bit additions and multiplications using parts of the round key
- XOR intermediate quarter blocks with the blocks from layer 1
- Exchange the inner quarter blocks

Output Transformation

- Done after the 8th round
- Exchange the inner quarter blocks
- Apply 16-bit additions and multiplications using parts of the round key
- Recombine the quarter blocks to the final result



Key Schedule

- 128 bit key
- 16-bit sub-keys used (52 times in total)
- Key simply divided into eight 16-bit sub-keys

Algorithm:

- Take the 8 first sub-keys
- Then rotate the key 25 bits to the left
- Then do it again

Decryption

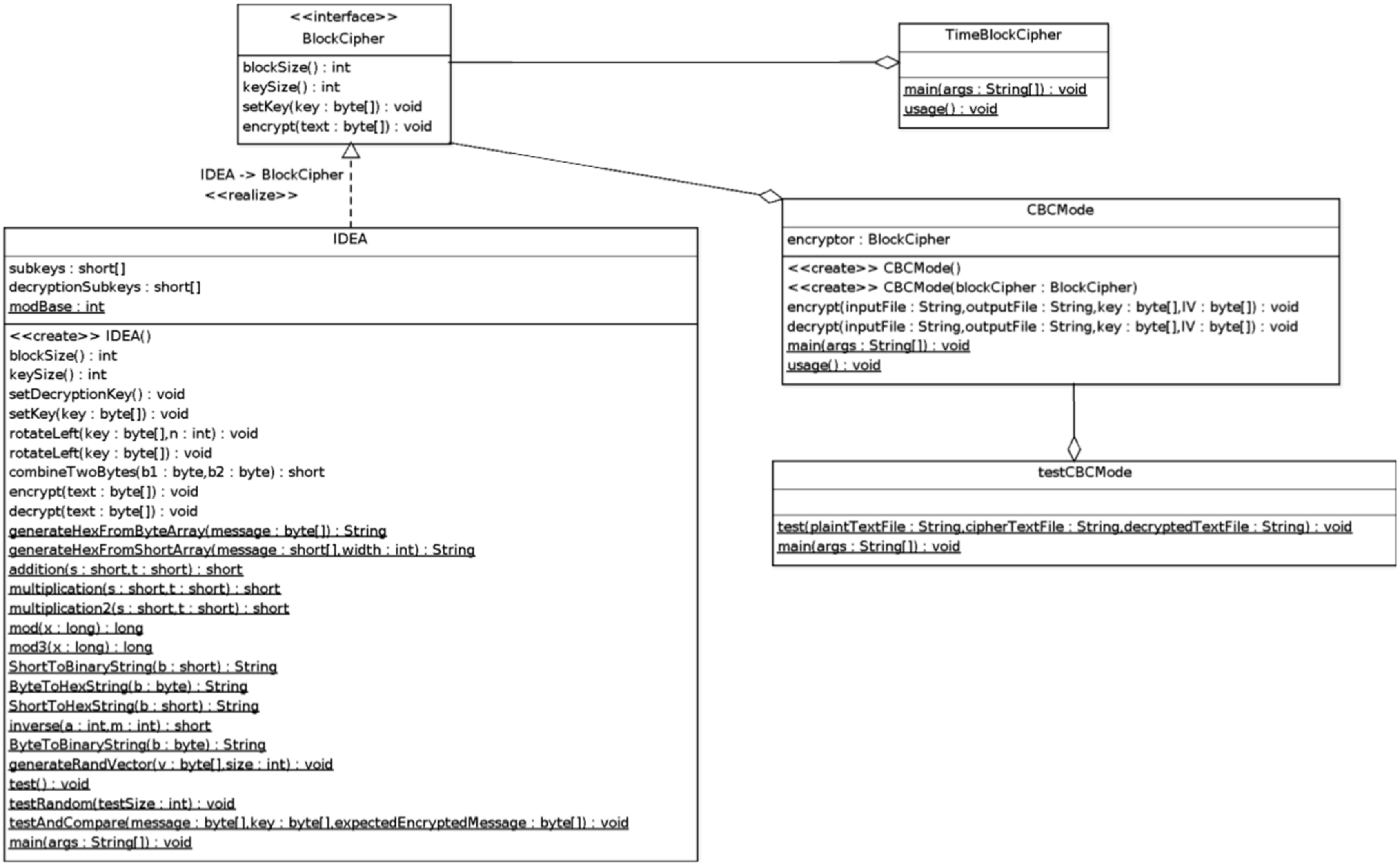
- Basically same as the encryption algorithm !
- Except the sub-keys
  - Schedule for decryption key is using combinations of multiplicative inverse, additive inverse, and original sub-keys
  - Example for round 1:
    - Z_1^{-1} $-Z_2$ $-Z_3$ Z_4^{-1} Z_5 Z_6

Security

- Considered as really secure
- Best attack can break IDEA reduced to 6 rounds (full IDEA = 8.5 rounds)

- But could be redesigned because of the very simple key schedule
- “Weak key” problem with too many 0-bits

Design of the software



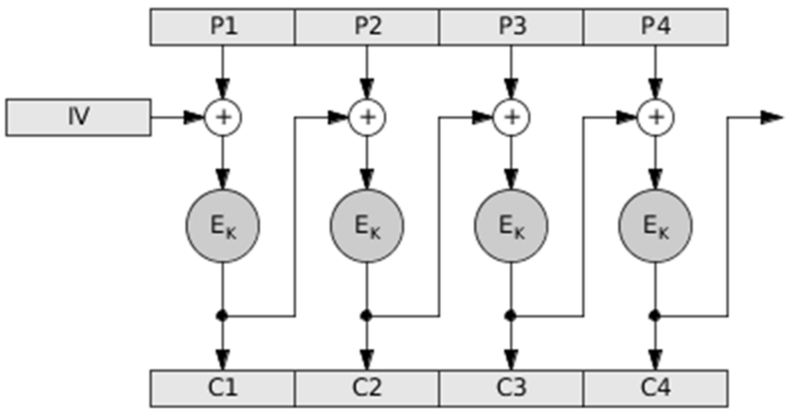
Design of the software – class IDEA

- setKey (byte[] key) : computes the 52 subkeys and the decryption subkeys (using mathematical methods)
- Encrypt(byte[] text) : applies the list of transformations using blocks and subkeys
- Decrypt(byte[] text) : applies the list of transformations using blocks and decryption subkeys

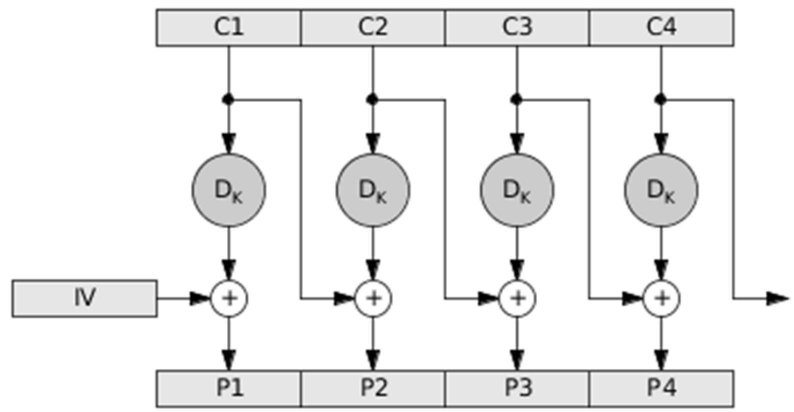
CBCMode

- Cipher Block Chaining Mode

Encryption :



Decryption:



- Read input file
- Add padding (10..0)
- Process encryption
- Write output file

- Reads input file
- Process decryption
- Remove padding
- Write output file

Usage

- Using CBCMode for file encryption/decryption

```
javaCBCMode <encrypt|decrypt> <inputFile> <outputFile>
<key_0> <key_1> ... <key_15> <IV_0> <IV_1> ... <IV_7>
```

- Example with keys=0 and IVs=0

  Encryption then decryption

Plaintext file	*Encrypted file*	*Decrypted file*
00 00 00 01 00 02 00 03	03 AA 00 88 02 EA FE 35	00 00 00 01 00 02 00 03
00 00 00 01 00 02 00 03	F6 33 03 44 0E 4F 04 C5	00 00 00 01 00 02 00 03
00 00 00 01 00 02 00 03	DC 98 2C B4 9D B5 92 60	00 00 00 01 00 02 00 03
00 00 00 01 00 02 00	ED 41 04 F4 1C 26 0A A5	00 00 00 01 00 02 00

Running time measurements & analysis

- Encrypt all-zero-byte plaintext block with all-zero-byte key 4.5 * 10^8 times in 313009msec (7.0 e-4msec / encryption)
- Most most frequently-executed methods:
  - IDEA.encrypt
  - IDEA.multiplication
  - IDEA.addition
- Most promising part for redesign: mod $2^{16} + 1$

Revised design & analysis

- Reduction modulo m = bt+c

  1. $q_0 \leftarrow x/b^t$, $r_0 \leftarrow x - q_0 b^t$, $r \leftarrow r_0$, $i \leftarrow 0$.
  2. If $q_i > 0$ do the following:
     2.1 $q_{i+1} \leftarrow q_i c/b^t$, $r_{i+1} \leftarrow q_i c - q_{i+1} b^t$.
     2.2 $i \leftarrow i + 1$, $r \leftarrow r - r_i$.
  4. If $q_i = 0$ return 1
  5. While $r \geq m$ do: $r \leftarrow r - m$.
  6. While $r < 0$ do: $r \leftarrow r + m$.
  7. Return(r).

- Replaces modulo operation with bitwise shift
- No need for multiplication because C=1

Further improvement

- Further simplification due to x being limited to 0xFFFE0001(0xFFFF * 0xFFFF)
- No need for loop

Algorithm Reduction modulo $m = 0x10000 + 1$.

INPUT: a base 2, positive integer x, and a modulus m = $0x10000 + 1$.

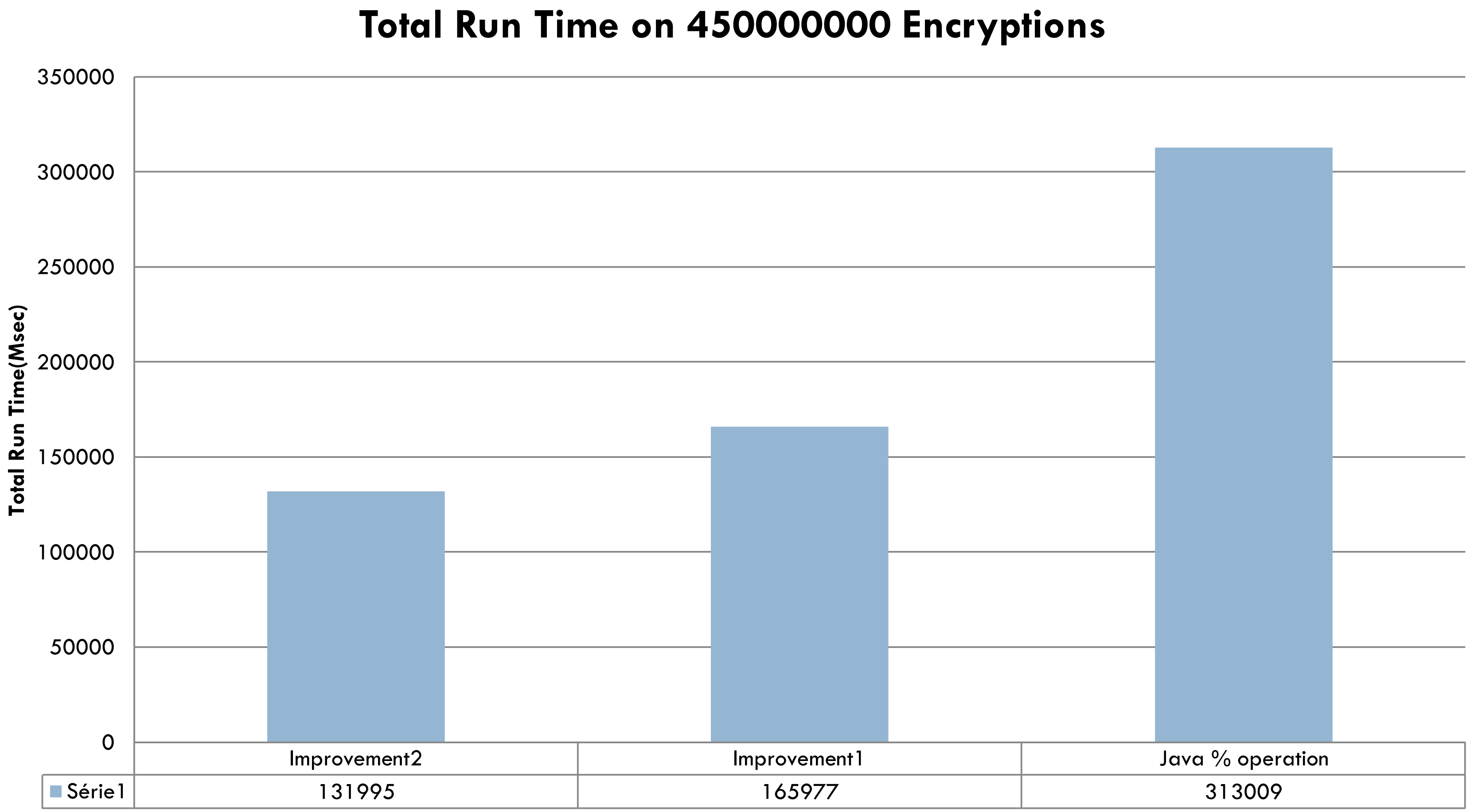
OUTPUT: r = x mod m.

1. r ← `((x & 0x000000000000FFFF) - ((x >> 16) & 0x000000000000FFFF)).`
2. If r = 0 return 1
3. While r ≥ m do: r ← r −m.
4. While r <0 do: r ← r+m.
5. Return(r).

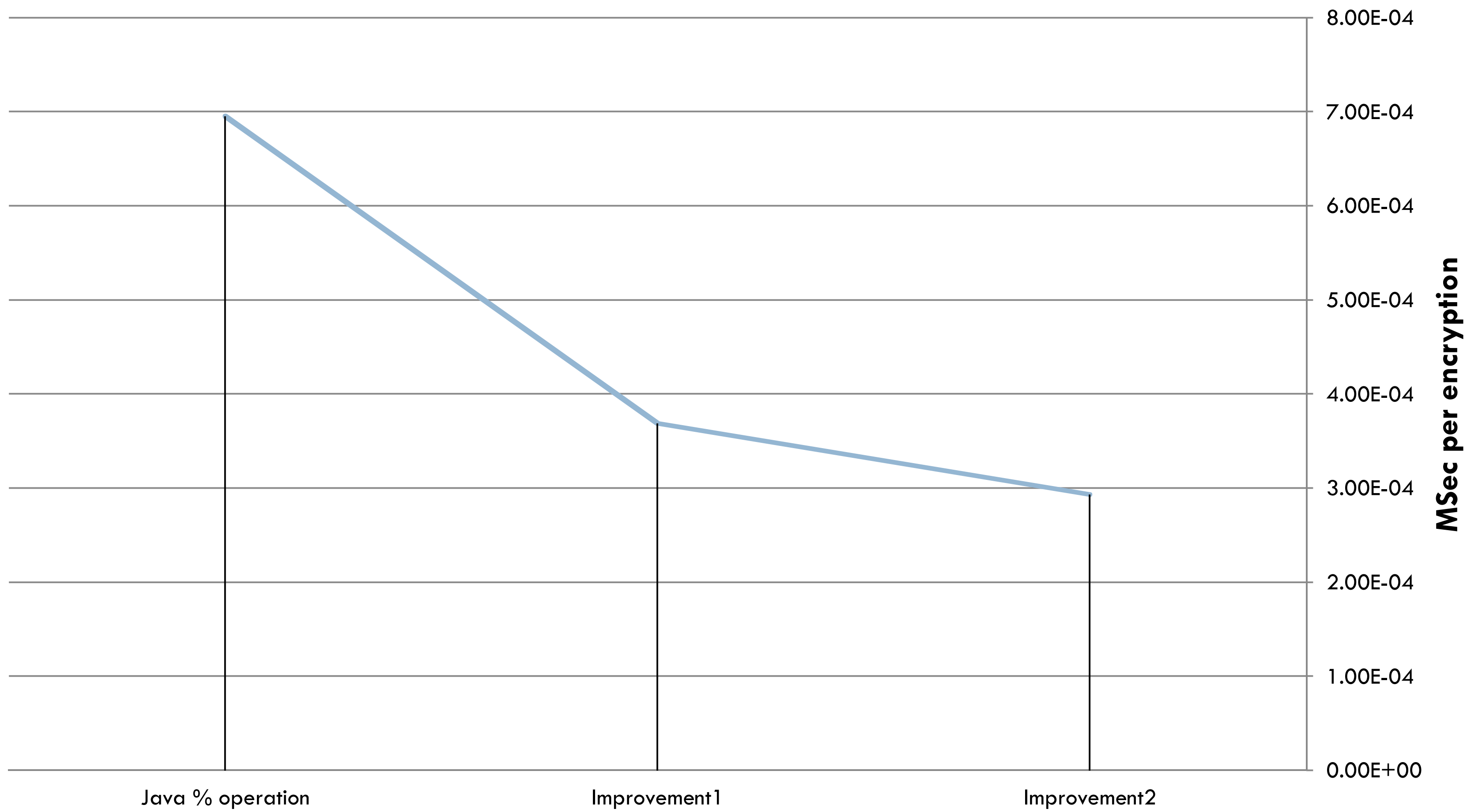
New measurements

- Modification 1
  - 165977msec (3.69 e-4msec / encryption)
  - 53.03% of the initial running time
- Modification 2
  - 131995msec (2.93 e-4msec / encryption)
  - 42.17% of the initial running time

Comparison


Total Run Time on 450000000 Encryptions
Total Run Time(Msec)
350000
300000
250000
200000
150000
100000
50000
0
Improvement2
Improvement1
Java % operation
Série1
131995
165977
313009

Comparison



8.00E-04
7.00E-04
6.00E-04
5.00E-04
4.00E-04
3.00E-04
2.00E-04
1.00E-04
0.00E+00
MSec per encryption
Java % operation
Improvement1
Improvement2

What we learned

- Specialized algorithm is better than generic
- Use Java profiling
- Optimize most frequently-executed piece of code
- Make piece of code run faster
- Remove *any* unnecessary operations

Possible future work

- Further optimize modular operations.
- Improve performance of file encryption by speeding up file I/O operations.
- Reduce number of intermediate computations in “encrypt” method.

References

- **Wikipedia article** http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm
- **United States Patent**

  http://www.google.com/patents/US5214703?printsec=abstract#v=onepage&q&f=false
- **RSA description of other block ciphers**

  http://www.rsa.com/rsalabs/node.asp?id=2254
- **Quadibloc article about IDEA Block Cipher (by John Savard) :**

  http://www.quadibloc.com/crypto/co040302.htm
- **-"Handbook of Applied Cryptography" (by Alfred J. Menezes)**

  www.cacr.math.uwaterloo.ca/hac