CAIRN 3: An FPGA Implementation of the Sieving Step with the Lattice Sieving

FUJITSU LTD.
Tetsuya Izu, Jun Kogure, *Takeshi Shimoyama

A part of this research is financially supported by a contract research with the National Institute of Information and Communications Technology (NICT), Japan
What is CAIRN3

Circuit Aided Integrated Relation Navigator

CAIRN 3
Special Purpose Hardware for Lattice Sieving
Outline of This Talk

1. Summary of the Integer Factoring by using a Hardware
2. Problems on the Implementation of the Lattice Sieving and Our Solutions
3. Security Evaluation of the RSA
Progress of the Development of the Sieving HW

- In 2005 (CAIRN1) [SHARCE2005]
  - Line Sieving
  - Implemented on DAPDNA2

- In 2006 (CAIRN2) [CHES2007]
  - Line Sieving and Relation checking
  - Implemented on FPGA × 2 and DAPDNA2

- In 2007 (CAIRN3) [This presentation]
  - Lattice Sieveing and Relation Checking
  - Implemented on FPGA × 3 and DAPDNA2
### Previous Works on the Integer Factoring HW

#### Progress of Researches

<table>
<thead>
<tr>
<th>Sieving Step</th>
<th>Relation Search</th>
<th>Check Step</th>
<th>Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NWS (2007)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Franke et al. (2005)</td>
<td>Gaj et al. (2006)</td>
<td></td>
</tr>
</tbody>
</table>

#### Generations

- **1st Generation**
  - BernsteinCircuit (2001)

- **2nd Generation**
  - LenstraCircuit (2002)

- **3rd Generation**
  - CAIRN1 (2005)
  - CAIRN2 (2006)
  - CAIRN3 (2007)
The Lattice Sieving Method is

- One of the procedure for the Sieving step in NFS
- Proposed by Pollard in 1991
- Using “Lattice Space” generated by Large Factor Base
- Used for Achieving the current Integer Factoring Records by using NFS
- The details of the implementation are written in few documents
Problems for the Lattice Sieving

There are 3-large problems for the Lattice Sieving.

1. Computation of the Lattice Base
   Need large amount of the multiplications and divisions

2. Management of the Lattice Base
   Most of all Lattice base (99%) is pass through the circuit without any operations if an easygoing way had been used.

3. Making the round of the Lattice Points
   Complicated Judgement of the Boundary of the Sieving Space
3. How to round the Lattice Points (1)

Lattice Sieving developed by Franke & Kleinjung

In the case that the lattice base satisfies “some conditions”, the all of the lattice points in the sieving space are treated by the following equation.

\[(c', d') = (c, d) + \begin{cases} (c_1, d_1) & c + c_1 \geq -HC \\ (c_2, d_2) & c + c_2 < HC \\ (c_1, d_1) + (c_2, d_2) & \text{otherwise} \end{cases}\]

3. How to sequence the Lattice Points (2)

Sieving Method

Memories 512KB in FPGA is assigned 2\(^{19}\)-sieving points (-2\(^{15}\) - 2\(^{15}\) - 1, 8*D - 8*D + 7) (D=0,…,8191)

1. For each i=0,…,11, execute the step 2
2. Execute the step 3-6 for each lattice Base in the data base SB\(_{ij}\) (j = D mod 2\(^i\)).
3. If d \geq 8D then goto step 6.
4. Add log(p) to the sieving point according to the coordinate (c,d). [Sieving]
5. Calculate new coordinate (c’, d’) by using the Franke&Kleinjung Method, and go to the step 3.
6. Store the lattice base [(c’,d’),(c\(_1\),d\(_1\)),(c\(_2\),d\(_2\))] to the tail of the data base Si\(_k\), where k = d/8 mod 2\(^i\). [Sorting]
1. Calculation of the Lattice Base

Input Parameter

(q, r) : Special-Q (One of the Factor base q: prime, r: integer smaller than q)
(a₁, b₁), (a₂, b₂) : Reduced base calculated by Special-Q
(p, t) : a Factor Base

Calculation

1. Translate (p, t) into the Lattice space generated by (a₁, b₁), (a₂, b₂)

\[ s = (a₁ + b₁t)(a₂ + b₂t)^{-1} \mod p \]

2. Calculate Lattice Base (α, β), (γ, δ) by using Franke & Kleijung method

\[
\begin{align*}
(i[0], j[0]) &= (-p, 0), (i[1], j[1]) = (s, 1), \\
Repeat & the following from k=1 \\
(i[k+1], j[k+1]) &= (i[k-1], j[k-1]) + a_k(i[k], j[k]), \\
a_k &= \left\lfloor \frac{|i[k-1]|}{|i[k]|} \right\rfloor, \\
If & |i[k]| < 1 \text{ and } |i[k-1]| \geq 1 \text{ then quit from the loop.} \\
k &= k+1,
\end{align*}
\]

Find the smallest “a” such that |i[k-1]| - a|i[k]| < 1

If k is even

\[
(\alpha, \beta) = (i[k-1], j[k-1]) + a(i[k], j[k])
\]

(γ, δ) = (i[k], j[k])

If k is odd

\[
(\alpha, \beta) = (i[k], j[k])
\]

(γ, δ) = (i[k-1], j[k-1]) + a(l[k], j[k])

Extended Binary GCD (a part)

Extended Lehmer GCD (a part)
Sieving Hardware

- Combination of the three kind of the devices
  1. Generator of the Lattice Base (FBASE)
  2. Lattice Siever (SIEVE)
  3. Checker of the relation (CHECKER)

- Flow of the calculation
Results of the Implementation

- Maximum input: 768-bit composite number
- Size in FPGA

<table>
<thead>
<tr>
<th></th>
<th>SLICE (%)</th>
<th>RAM(%)</th>
<th>LUT(%)</th>
<th>Register(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBASE</td>
<td>99.998%</td>
<td>89.0%</td>
<td>80.1%</td>
<td>48.8%</td>
</tr>
<tr>
<td>SIEVE</td>
<td>40.9%</td>
<td>98.8%</td>
<td>32.1%</td>
<td>19.6%</td>
</tr>
<tr>
<td>CHECKER</td>
<td>78.0%</td>
<td>40.0%</td>
<td>45.0%</td>
<td>42.0%</td>
</tr>
<tr>
<td>Total</td>
<td>89.088%</td>
<td>336</td>
<td>178,176</td>
<td>178,176</td>
</tr>
</tbody>
</table>

- Throughput
  - C128
    - Lattice Base: FPGA, Time: 0.0992 sec
    - Sending Lattice Base: DIO, Time: 0.1080 sec
    - Initial Setting: CPU, Time: 7.959 sec
    - Sieving: FPGA, Time: 10.915 sec
    - Sending Relation: EtherNet, Time: 0.237 sec

  - RSA768
    - Lattice Base: FPGA, Time: 2.75 sec
    - Sending Lattice Base: DIO, Time: 2.40 sec
    - Initial Setting: CPU, Time: 61.49 sec
    - Sieving: FPGA, Time: 190.43 sec
    - Sending Relation: EtherNet, Time: 0.044 sec
Security Evaluation of RSA

- Estimation for factoring 768 bit RSA key
  - Timing for one relation = 3.920 sec.
  - The number of requirement of the relations = $2.17 \times 10^9$
  - $3.920 \text{ sec.} \times (2.17 \times 10^9) = \text{About } 270 \text{ years}$

- Comparison with the previous HW in 2005
  - In the case of RSA768, this HW achieved 38 times good performance compared with the previous one\(^{(※1)}\).

<table>
<thead>
<tr>
<th></th>
<th>Line Sieve</th>
<th>Lattice Sieve</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation ratio</td>
<td>1</td>
<td>23.43</td>
<td>By effect of Line Sieve→Lattice Sieve</td>
</tr>
<tr>
<td>Throughput (ms)</td>
<td>49.92</td>
<td>30.75</td>
<td>By Minor Improvement</td>
</tr>
<tr>
<td>Comparison</td>
<td>1</td>
<td>38.04</td>
<td>$23.43 \times (49.92/30.75)$</td>
</tr>
</tbody>
</table>

- Comparison with the Software
  - According to the Software evaluation\(^{(※2)}\), it will take 1108 years for one PC (AMD Opteron). Then, the CAIRN3 is about 4.1 times faster than PC.

※1 See also CHES2007 (2007.9.13)
※2 From the Kleinjung Report (CRYPTREC Report 2006)
Conclusion

- Implementation of CAIRN3
  - It succeeded in the development of the Relation Search HW CAIRN3 by using Lattice Sieving (The 1st in the World!)
  - In the case of RSA768, CAIRN3 achieved 38 times good performance compared with CAIRN2 (World Record!)

- Security Evaluation of RSA
  - We estimated that it will take about 270 years for factoring the 768-bit RSA key by using ond CAIRN3.
  - For 1024-bit RSA, we concluded that the development of the sieving hardware is almost impossible in a current technologies we have, because it is necessary to solve the following several difficult problems.
    - Require at least 19.4GB memory and large-scale memory control circuit,
    - Require High-speed data transfer circuit between each devices,
    - Require Large-scale data sorting circuit, etc…
  - The 1024-bit RSA would not become insecure for several years, even if we have used HW which is based on a current technologies and practical resources.
THE POSSIBILITIES ARE INFINITE