Implementation of the PRIMATEs family of Authenticated Ciphers

Final Presentation

ECE 646 Fall 2014

Ahmed Ferozpuria
The PRIMATEs Family

• APE, GIBBON, and HANUMAN
• Targeted for lightweight devices
• Duplex Sponge Construction
• Permutation ($p_1$ – $p_4$ : 6 or 12 rounds)
  – SubElements (SE)
  – ShiftRows (SR)
  – MixColumns (MC)
  – Constant Addition (CA) – 4 LFSR Initialization values
  – Inverse mode only required for APE
Sponge Construction

Figure 1 - Overview of sponge construction [1]
Duplex Sponge Construction

Figure 2 - Overview of duplex sponge construction [1], also used by APE.
PRIMATEs Sponge Construction

Figure 3 - Modified duplex sponge construction [2], used by HANUMAN and GIBBON

- PRIMATEs sponge construction facilitates Release of Unverified Plaintext (RUP)
- Useful for lightweight devices
Figure 4 – The $5 \times 5$ matrix representation of the PRIMATE state

- Either $5 \times 8$ or $7 \times 8 = 200$ or $280$ bit internal state
- Output taken from $V_r$
- The rest of the matrix is called the capacity, $V_c$
The PRIMATE permutation

No key scheduling
The PRIMATE permutation with inverse mode

Only used for APE decryption
Top Level Overview
Inside Cipher Core

Primate Cipher (PC)

Data Processor (DP)

clk reset

Do
PDI
SDI

DATA

SYNC

pdi_ready
sdi_ready
do_ready
pdi_read
sdi_read
do_write
error
Inside Primate Cipher (level 1)
Workflow Process

- Eclipse - test vectors
- ISE Webpack and ISim - waveform
- Test vectors run in sequence for different cases of (AD, M)
  - (0,0) \(\rightarrow\) (1,0) \(\rightarrow\) (2,0) \(\rightarrow\) (0,1)
    \(\rightarrow\) (0,2) \(\rightarrow\) (1,1) \(\rightarrow\) (2,2)
- Helpful to debug FSM
- For each case:
  - Check sequences of data/algorithm flow
  - Verify internal state output
- Upon successful case verification move to next case OR fix and retest previous cases.
Comparison & Analysis

• Target FPGA: Virtex 6 XC6VLX75T FF784 (-3)
• FPGA utilization analysis: latency, throughput, area, throughput/area, timing, max freq
• Hardware / Software execution comparison
• Benchmarking on ATHENa
General Datapath Approach

\[ (y, T_o, p_i) = f_n(K, IV, X, T_i, p_o) \]

\[ \text{Registered Transformation} \]

\[ \text{done} \]

\[ \text{ctrl} \]

\[ p_o \]

Algorithm 1: \( E_K(N, A, M) \)

Input: \( K \in \mathbb{C}, N \in \mathbb{C}^t, A \in \{0, 1\}^s, M \in \{0, 1\}^t \)
Output: \( C \in \{0, 1\}^s, T \in \mathbb{C} \)

1. \( V \leftarrow 0^t \parallel K \)
3. for \( i = 1 \) to \( y \) do
   4. \( V \leftarrow p_1(N[i] \oplus V_r \parallel V_c) \)
4. end
5. if \( A \neq \emptyset \) then
   7. \( A[u] \leftarrow A[u] \parallel 10^s \)
   8. for \( i = 1 \) to \( u \) do
      9. \( V \leftarrow p_1(A[i] \oplus V_r \parallel V_c) \)
   10. end
11. end
12. \( V \leftarrow V \oplus (0^{b-1} \parallel 1) \)
14. \( l \leftarrow |M[w]| \)
15. \( M[w] \leftarrow M[w] \parallel 10^s \)
16. for \( i = 1 \) to \( w \) do
    17. \( V \leftarrow p_1(M[i] \oplus V_r \parallel V_c) \)
    18. \( C[i] \leftarrow V_r \)
19. end
21. \( C \leftarrow C \parallel [C[w-1]]_t \)
22. \( C \leftarrow C \parallel C[w] \)
23. \( T \leftarrow V_r \oplus K \)
24. return \((C, T)\)

APE Algorithm [2]
Datapath for each PRIMATE

APE

GIBBON

HANUMAN
Datapath FPGA utilization

<table>
<thead>
<tr>
<th></th>
<th>HANUMAN</th>
<th>GIBBON</th>
<th>APE(E)</th>
<th>APE(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTs</td>
<td>1046</td>
<td>1043</td>
<td>1366</td>
<td>2309</td>
</tr>
<tr>
<td>Flip-Flops</td>
<td>209</td>
<td>209</td>
<td>409</td>
<td>414</td>
</tr>
</tbody>
</table>

Note: APE (D) ≈ APE(E) + HANUMAN = 1366 + 1046 = 2412
General Control Approach
Flowchart for each PRIMATE (coming soon)

APE        GIBBON        HANUMAN
Control FPGA utilization

<table>
<thead>
<tr>
<th></th>
<th>HANUMAN</th>
<th>GIBBON</th>
<th>APE(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>States</td>
<td>12</td>
<td>22</td>
<td>15</td>
</tr>
<tr>
<td>Transitions</td>
<td>25</td>
<td>41</td>
<td>30</td>
</tr>
<tr>
<td>Inputs</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Outputs</td>
<td>6</td>
<td>8</td>
<td>7</td>
</tr>
</tbody>
</table>
# Overall FPGA utilization (Top Level)

<table>
<thead>
<tr>
<th></th>
<th>HANUMAN</th>
<th>GIBBON</th>
<th>APE (E)</th>
<th>APE (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTs</td>
<td>1,918</td>
<td>1,956</td>
<td>2,452</td>
<td>3,864</td>
</tr>
<tr>
<td>Flip-Flops</td>
<td>771</td>
<td>775</td>
<td>1,134</td>
<td>1,138</td>
</tr>
<tr>
<td>Slices</td>
<td>593</td>
<td>593</td>
<td>702</td>
<td>1,298</td>
</tr>
<tr>
<td>Max Freq (MHz)</td>
<td><strong>133</strong></td>
<td>160</td>
<td>143</td>
<td>137</td>
</tr>
</tbody>
</table>

Bonded IOBs: 135
## Hardware Timing Analysis

<table>
<thead>
<tr>
<th></th>
<th>HANUMAN</th>
<th>GIBBON</th>
<th>APE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD Throughput</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Message/Cipher</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td>Throughput</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Clock Cycles Overview

<table>
<thead>
<tr>
<th></th>
<th>Key Scheduling</th>
<th>Authenticated Encrypt</th>
<th>Authenticated Decrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td>APE</td>
<td>0</td>
<td>$12(n^+ + m + k) + 1 + 1$</td>
<td>$12(n^+ + m + k) + 1 + 12$</td>
</tr>
<tr>
<td>HANUMAN</td>
<td>0</td>
<td>$12(n^+ + m) + 12$</td>
<td>$12(n^+ + m) + 12$</td>
</tr>
<tr>
<td>GIBBON</td>
<td>0</td>
<td>$6(n^+ + m) + 12 + 1 + 6 + 12$</td>
<td>$6(n^+ + m) + 12 + 1 + 6 + 12$</td>
</tr>
</tbody>
</table>

- $k$ - the number of IV blocks
- $n$ – the number of plaintext/ciphertext blocks
- $m$ - the number of associated data blocks
## Benchmarking Results

<table>
<thead>
<tr>
<th></th>
<th>HANUMAN</th>
<th>GIBBON</th>
<th>APE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


Software Execution

- Using MSP430F5229 Launchpad
- Execute basic encryption / decryption of 10 bytes
- Use stopwatch to measure elapsed time
## Software Timing Results

<table>
<thead>
<tr>
<th></th>
<th>HANUMAN</th>
<th>GIBBON</th>
<th>APE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption (s)</td>
<td>10.23</td>
<td>9.47</td>
<td>14.16</td>
</tr>
<tr>
<td>Decrypt (s)</td>
<td>9.77</td>
<td>9.10</td>
<td>13.68</td>
</tr>
</tbody>
</table>
Conclusions

- APE has the largest FPGA utilization
- Hanuman has the lowest max frequency
- GIBBON has the largest algorithm flow control
- Hardware Implementation offers a speed up of about __ over the MSP430F5229 execution.
Difficulties

• Eclipse doesn’t like new projects 😊
• Determining a good work flow strategy
  – Save your waveforms!
  – A change that fixes one problem may cause many others
• Code Composer Studio required some code changes
  – .cpp -> .c
  – for (int i=0; ;) vs for(i=0; ;)
• Design document inaccurate (ie v1.0 vs v1.02)
  – Refer to c code
Future Work

- Minimize algorithm flow control
- Loop unrolling effects on throughput and latency
- Implementation on FPGA
Questions

References:

2. PRIMATEs v1.02, submission to the CAESAR competition