HyRAL
(Hybrid Randomization Algorithm)

Self-Evaluation Report

Kouzou Hirata
Laurel Intelligent Systems Co., Ltd,
e-ver2
Table of Contents

1. Design philosophy ................................................................. 2
2. HyRAL configuration based on our design philosophy ............ 2
   2.1 S-box .................................................................................. 2
   2.2 MDS conversion ................................................................. 2
   2.3 MDS line method ............................................................... 3
   2.4 Basic functions ($f_i$ function) .............................................. 4
   2.5 Expanded functions $G_1, G_2, F_1, F_2$ .............................. 5
   2.6 Overall flow ...................................................................... 5
3. Foundation theories and technologies .................................. 7
4. Safety evaluations ............................................................... 7
5. Evaluation of software integration ........................................ 7
   5.1 Resource usage ................................................................. 7
   5.2 Language ........................................................................... 7
   5.3 Evaluation ................................................................. 8
      5.3.1 Endian evaluation ...................................................... 8
      5.3.2 Multiple platform evaluation ...................................... 8
      5.3.3 Speed evaluations ................................................... 8
      5.3.4 Comparison with AES ................................................ 9
6. Evaluation of hardware integration ....................................... 10
   6.1 Evaluation environment .................................................... 10
      6.1.1 FPGA ............................................................................ 10
      6.1.2 Simulation environment .............................................. 10
      6.1.3 Compilation and layout environment ............................ 10
   6.2 Evaluation results ............................................................. 10
      6.2.1 Logic verification performed on Model-Sim .................... 10
      6.2.2 Process time ............................................................. 10
      6.2.3 Compilation and layout results were obtained using ISE made by Xilinx............................ 12
      6.2.4 Synthesis Report ......................................................... 14
      6.2.5 Translation Report ...................................................... 14
      6.2.6 Static Timing Report .................................................. 14
   6.3 Conclusion ................................................................. 14
7. Side channel attacks ............................................................. 15
   7.1 Protection against cache attacks ......................................... 15
   7.2 Protection against SPA and DPA attacks .............................. 16
1. Design philosophy

An important premise in designing HyRAL was to achieve sufficient robustness against known decoding technologies. It was also designed so that its coding technology can be expanded for use in the following three applications:

(1) Achieve both coding and authorization based on a calculation using one key and one block coding.
Communications and files require both confidentiality and integrity. In order to achieve both of these objectives, currently recommended methods require calculations that are equivalent to two block codings. From the standpoint of performance, these tasks should preferably be achieved in a single calculation.

(2) Use of hash functions based on block coding technology.
As for hash functions, an SHA-3 will be chosen in the US and adopted as a standard in the near future. Widely used hash functions in the past consisted of unique algorithms that were different from block coding, and while they have delivered better performance compared to block coding, they no longer provide sufficient levels of safety due to progress being made in decoding technologies. This is why a public tender has been made for SHA-3. Regardless of the type of algorithm chosen as the SHA-3, we believe that—as long as the same level of performance is achieved—any application of block coding technology should fulfill the objectives.

(3) Multi-block coding
Variable-length block coding has emerged as a new requirement. This design philosophy requires support for variable-length block coding, as opposed to methods that use repetitions of block codings based on blocks of fixed lengths.

Based on these three points, we also focused our attention on developing a generally simple method for particular programs. Most of the programming can be done using table searches and word-based (4 byte) XOR operations.

2. HyRAL configuration based on our design philosophy

2.1 S-box
S-box represents the minimum unit of nonlinear portions in block coding. An S-box in HyRAL is the same as in AES and is not original in this sense.
However, while AES uses the 1/x function as the S layer, after which a linear process is performed as the P layer, and the result of this process is used as the S-box, the S-box in HyRAL is generated by performing the P layer first and then using the 1/x function.
Other than this, HyRAL's maximum linear and differential probabilities are both $2^{-6}$.

2.2 MDS conversion
Per-byte conversions are performed via S-box as the minimum nonlinear process, and an MDS conversion is performed to linearly convert this in units of 4-bytes (32 bits). This is the same idea behind the "Mix column" in AES.
The most important characteristic of HyRAL is that unbalanced MDS conversion is used for MDS conversion.
Most MDS conversions used in coding technologies are balanced. "x" is said to be balanced when $x_1 = x_2 = x_3 = x_4$ is true for 4-byte variables $x = (x_1, x_2, x_3, x_4)$ entered in MDS, and means that when the input is balanced, the output is also balanced.
HyRAL uses noncyclic MDS lines and is therefore not balanced. Because in most block coding schemes, the value entered in MDS undergoes an XOR process with a subkey, referred to as the step key, and converted at the S layer, this lack of balance does not represent an immediate vulnerability. This is because, numerous subkeys are used in the entirety of the block coding and not all of them are balanced. If all step keys are balanced, any balanced plain text entered in the block coding will always result in balanced coded text.

As described above, while the absence of balance may not represent a vulnerability, HyRAL uses noncyclic MDS lines to take account of the possibility that this feature may be exploited by currently unknown methods of attack.

2.3 MDS line method
The fact that 4x4 lines are MDS lines is equivalent to the fact that an arbitrary ixi minor \((i = 1,2,3,4)\) will be non-zero.
The 4x4 MDS line for HyRAL was obtained by way discovery using the following steps:

1. Generate a holomorphic 4x4 line.
2. Check to see if all of the fourth bytes of the output become non-zero if only the first byte of the input is non-zero. If it fails to meet this condition, go to step (1).
3. Check to see if all of the third bytes of the output become non-zero if only the second byte of the input is non-zero. In other words, if this kind of entry is properly chosen, only the first byte can be rendered zero. If it fails to meet this condition, go to step (1).
4. Check to see if the first and second bytes of the output can be rendered zero if an input is selected where only the third byte of the input is non-zero. If it fails to meet this condition, go to step (1).

If a condition was not met in any of these steps, a basic operation was performed on the original formal line to avoid this state of unmet conditions. As these steps were carried out based on observation and discovery, a calculator was used for verification by performing a non-zero inspection of the ixi minor \((i = 1,2,3,4)\) for the final MDS line candidate.
2.4 Basic functions ($f_i$ function)

HyRAL uses unbalanced MDS lines, such as that mentioned in paragraph 2.2, which are defined as $f_i$ functions.

![Diagram of unbalanced MDS line]

The unbalanced MDS line is a synthesis of the following two lines.

$$
\begin{pmatrix}
    c_0 \\
    c_1 \\
    c_2 \\
    c_3
\end{pmatrix}
= 
\begin{pmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 3 \\
    0 & 0 & 1 & 3 \\
    2 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    b_0 \\
    b_1 \\
    b_2 \\
    b_3
\end{pmatrix}
= 
\begin{pmatrix}
    3 & 3 & 2 & 1 \\
    2 & 4 & 1 & 1 \\
    4 & 5 & 2 & 1 \\
    1 & 2 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
    a_0 \\
    a_1 \\
    a_2 \\
    a_3
\end{pmatrix}
\begin{pmatrix}
    3 & 3 & 2 & 1 \\
    1 & 2 & 2 & 2 \\
    7 & 3 & 1 & 2 \\
    7 & 4 & 5 & 3
\end{pmatrix}
\begin{pmatrix}
    a_0 \\
    a_1 \\
    a_2 \\
    a_3
\end{pmatrix}
$$

The $CST_i$ calculation shown in Fig. 1 is used in situations where $a0$, $a1$, $a2$, and $a3$ are all 0x00. $CST_i$ will be included in the table and therefore does not need to be searched.

The $f_i$ function consists of 4 table searches and 3 XOR operations.

Even when a balanced MDS line is used, a constant (CST) can be added to the output to disrupt the balance, and some proposals do use similar methods to disrupt the balance. Even in such cases however, the balance...
in differential cannot be disrupted.
If this basic MDS conversion is balanced, this will pose problems for using this function in hash functions. Because HyRAL's MDS conversion is unbalanced, input transpositions of up to 8 $f$ functions can be utilized as the $f_i$ function ($i = 1$ through 8).
This is one element that makes its expansion to other coding modes possible based on block coding mentioned in the "Design philosophy" paragraph.

2.5 Expanded functions $G_1$, $G_2$, $F_1$, $F_2$
Each of the expanded functions using $f_i$ functions consist of 4 rounds.
The $G_1$ and $G_2$ functions are of the Feistel type with 3 inputs and 1 output, and the $f_i$ function is calculated once for each round.
The $F_1$ and $F_2$ functions are of the Feistel type with 2 inputs and 1 output, and the $f_i$ function is calculated twice for each round, and are vertically connected.
Each of these four functions have a inverse function, each of which have 128-bit block coding functions of their own.
HyRAL aims to achieve coding through a simple vertical connection of these 4 functions. If the key length is 128 bits, it takes on a $G_1 \rightarrow F_2 \rightarrow F_2 \rightarrow F_1 \rightarrow G_2$ configuration, and if the key length is 192 or 256 bits, it takes on a $G_1 \rightarrow F_2 \rightarrow F_2 \rightarrow F_2 \rightarrow F_1 \rightarrow F_1 \rightarrow G_2$ configuration. With a key length of 128 bits, it is referred to as the Single Key mode, and with a key length of 192 or 256 bits, it is called the Double Key mode.
Unlike the two-part Feistel type, decoding uses the inverse functions of each, and the subkey is used in the opposite sequence as that used in coding.

2.6 Overall flow
Fig. 2 on the following page shows the overall flow of the Single Key mode and the relationship between the round key ($RK$) and intermediate key ($IK$).
The $\Delta P_i$ of each round shown in Fig. 2 indicates the output of the $f_i$ function. $\Delta P_i$ is XOR-processed in the sequence shown in the drawing so that coding proceeds one stage at a time. Of the $\Delta P_i$ values, $\Delta P_{20}$ through $\Delta P_{20}$ are the output results of the vertical connection of the $f_i$ function. The $f_i$ functions used for each of the rounds are $f_1$ through $f_8$.
The key entered prior to the second process of the vertical connection of the $f_i$ function is the intermediate key ($IK$) of which 32 bits are used for each round.
The $f_i$ function is a 32-bit input/32-bit output coding block and we have denoted this as $\Delta P_i$ to indicate that it is the result of permutation.
While this block coding shows this simply as $\Delta P_i$, when used as an expanded mode of HyRAL block coding, this $\Delta P_i$ carries a significant meaning as internal chain information.
HyRAL block coding uses a minimum non-linear S-box with a minimum unit of 8 bits, and comprises 8 types of block coding of units of 32 bits. In other words, coding is achieved by using 4 types of 128-bit block coding which have been expanded to 128-bit block coding from block coding of units of 32 bits.
And the most important characteristic is that it has a permutation construction that does not exist in the key.
Flow of 32-bit data: →
Flow of 128-bit data: ↔
XOR of 32-bit data: ⊕
XOR of 128-bit data: ⊗

Fig. 2
3. Foundation theories and technologies

The S-box is designed as an 8-bit S-box, and ensures maximum differential and linear probabilities of \(2^{-6}\).

It has been made using an \(x^{-1}\) function after performing a linear conversion based on the \(x^{-1}\) function (\(\text{MOD}^{8}+x^4+x^3+x+1\)). The differential and linear probabilities of the \(x^{-1}\) function is maintained in the linear layer, allowing us to eliminate fixed points and polar points.

MDS conversion is used to realize a minimum number of branches of 5 when it is handled for units of 32 bits with the S-box being the non-linear permutation. Many coding technologies have used MDS conversion and all can be said to have been balanced just as in AES.

In a balanced MDS conversion, the balance is disrupted using only the key.

HyRAL uses unbalanced MDS conversion.

The decision to use this strategy was made after verifying whether the MDS conversion ultimately used was correct. This was verified by using a calculator to check for its non-zero property of \(\text{minor } i,i+1,\ldots,i+4\) (i =1,2,3,4).

"5" branches were secured for MDS conversion, and an unbalanced MDS conversion was developed and used.

4. Safety evaluations

In the HyRAL announced earlier\(^{[1]}\), safe upper bound for differential and linear probabilities were not obtained\(^{[2]}\). We therefore made improvements to ensure HyRAL's safety. We also asked a group of experts to conduct a detailed evaluation of its differential and linear robustness. As a result, we were able to show that the upper bound for linear and differential characteristic probability were \(2^{-228}\) and \(2^{-222}\) respectively with a 128-bit secret key, and \(2^{-318}\) and \(2^{-342}\) respectively with a 192-bit or 256-bit secret key.

Maximum upper bound for linear\(^{[3]}\) and differential\(^{[4]}\) characteristic probability based on 8-bit truncation were announced, as shown above, by the Tokyo University of Science, Kaneko lab at the SCIS2010 Symposium on Cryptography and Information Security held in January 2010. Going forward, we plan to make announcements relating to robustness against other forms of attack.

\(^{[1]} \) "HyRAL16 Symmetric-Key Coding" Shingakugiho, p. 29, ISEC 2006-76 (2006-09): Kouzou Hirata


\(^{[3]} \) Igarashi, Takagi, Kaneko: Security evaluation of HyRAL® against linear cryptanalysis SCIS2010, 1D1-3

\(^{[4]} \) Takagi, Igarashi, Kaneko: Security evaluation of HyRAL® against Differential Attack SCIS2010, 1D1-2

5. Evaluation of software integration

5.1 Resource usage

Amount of code: 99KB [Source: 93KB, Header: 6KB]

Work area: 8,264 bytes maximum [8,000 bytes minimum]

* The maximum and minimum values for work area size are shown as these vary depending on the compilation option used.

5.2 Language

ANSI C
5.3 Evaluation

<table>
<thead>
<tr>
<th>Item number</th>
<th>Description of evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.1</td>
<td>Endian evaluation</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Multiple platform evaluation</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Speed evaluation</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Comparison with AES</td>
</tr>
</tbody>
</table>

5.3.1 Endian evaluation

Evaluations were carried out using both little and big Endian machines.

We were able to verify that calculated results were the same as that obtained from the tools "HyRAL Encryption_20091214.xls" and "HyRAL Decryption_20091214.xls".

The following table shows the machines and OSs on which evaluations were carried out.

<table>
<thead>
<tr>
<th>Endian</th>
<th>CPU</th>
<th>Memory</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little</td>
<td>Intel CoreDuo 1.8GHz</td>
<td>512MB</td>
<td>Mac OS X 10.5 Leopard</td>
</tr>
<tr>
<td>Big</td>
<td>PowerPC G4 867MHz (dual)</td>
<td>768MB</td>
<td>Mac OS X 10.4 Tiger</td>
</tr>
</tbody>
</table>

* Endian support can be achieved by specifying Endian in the compilation option.

5.3.2 Multiple platform evaluation

We evaluated operations on multiple platforms.

We were able to verify that calculated results were the same as that obtained from the tools "HyRAL Encryption_20091214.xls" and "HyRAL Decryption_20091214.xls".

The following table shows the machines and OSs on which evaluations were carried out.

<table>
<thead>
<tr>
<th>Endian</th>
<th>CPU</th>
<th>Memory</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little</td>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2GB</td>
<td>Windows XP Pro SP3</td>
</tr>
<tr>
<td>Little</td>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1GB</td>
<td>Windows XP Pro SP3</td>
</tr>
<tr>
<td>Little</td>
<td>Intel Core Duo 1.8GHz</td>
<td>512MB</td>
<td>Mac OS X 10.5 Leopard</td>
</tr>
<tr>
<td>Big</td>
<td>PowerPC G4 867MHz (dual)</td>
<td>768MB</td>
<td>Mac OS X 10.4 Tiger</td>
</tr>
</tbody>
</table>

5.3.3 Speed evaluations

- Method of speed evaluation

1. The processor that is able to execute the specified thread is (exclusively) specified (Supports dual CPUs) using the SetThreadAffinityMask.
2. The clock is obtained (c0) using the assembler command rdtsc.
3. The processes to be measured (key scheduling, coding process, decoding process) are repeated a given number of times.
4. The clock is obtained (c1) using the assembler command rdtsc.
5. The clock time required to perform the processes is calculated as c1 - c0.
6. The above process is repeated for 10 cases and the smallest clock value is output.

<table>
<thead>
<tr>
<th>Actual measurement in Hz</th>
<th>CPU Hz as measured on the machine used for evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock</td>
<td>c1 - c0</td>
</tr>
<tr>
<td>Loop</td>
<td>The number of times processes were repeated for the evaluation</td>
</tr>
<tr>
<td>Clock/1Block(1keyGen)</td>
<td>ClockCycles elapsed for a single block are measured: Clock/Loops</td>
</tr>
<tr>
<td>Sec/1Block(1KeyGen)</td>
<td>The number of seconds elapsed for a single block: (Clock/Loops)/Actual measurement in Hz</td>
</tr>
</tbody>
</table>
### Method of speed evaluation

The following table shows the machines and OSs on which evaluations were carried out.

<table>
<thead>
<tr>
<th>Endian</th>
<th>CPU</th>
<th>Memory</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little</td>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2GB</td>
<td>Windows XP Pro SP3</td>
</tr>
<tr>
<td>Little</td>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1GB</td>
<td>Windows XP Pro SP3</td>
</tr>
</tbody>
</table>

### Speed evaluation results

<<Key generation [128-bit]>>

<table>
<thead>
<tr>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1KeyGen</th>
<th>Sec/1keyGen</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>614634370</td>
<td>540.491574000</td>
<td>0.000000230</td>
<td>1000000</td>
</tr>
<tr>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>536723556</td>
<td>533.621363000</td>
<td>0.000000289</td>
<td>1000000</td>
</tr>
</tbody>
</table>

<<Coding process [128-bit]>>

<table>
<thead>
<tr>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1Block</th>
<th>Sec/1Block</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>676443936</td>
<td>668.277442000</td>
<td>0.000000284</td>
<td>1000000</td>
</tr>
<tr>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>659460428</td>
<td>656.071862000</td>
<td>0.000000356</td>
<td>1000000</td>
</tr>
</tbody>
</table>

<<Decoding process [128-bit]>>

<table>
<thead>
<tr>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1Block</th>
<th>Sec/1Block</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>662740414</td>
<td>662.740414000</td>
<td>0.000000282</td>
<td>1000000</td>
</tr>
<tr>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>654356835</td>
<td>653.824521000</td>
<td>0.000000355</td>
<td>1000000</td>
</tr>
</tbody>
</table>

<<Key generation [256-bit]>>

<table>
<thead>
<tr>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1KeyGen</th>
<th>Sec/1keyGen</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>1337548856</td>
<td>1212.252034000</td>
<td>0.000000515</td>
<td>1000000</td>
</tr>
<tr>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>1153811271</td>
<td>1146.935333000</td>
<td>0.000000622</td>
<td>1000000</td>
</tr>
</tbody>
</table>

<<Coding process [256-bit]>>

<table>
<thead>
<tr>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1Block</th>
<th>Sec/1Block</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>891542778</td>
<td>891.542778000</td>
<td>0.000000379</td>
<td>1000000</td>
</tr>
<tr>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>887742704</td>
<td>886.610207000</td>
<td>0.000000481</td>
<td>1000000</td>
</tr>
</tbody>
</table>

<<Decoding process [256-bit]>>

<table>
<thead>
<tr>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1Block</th>
<th>Sec/1Block</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>906877224</td>
<td>900.269290000</td>
<td>0.000000488</td>
<td>1000000</td>
</tr>
<tr>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>905184049</td>
<td>900.269290000</td>
<td>0.000000488</td>
<td>1000000</td>
</tr>
</tbody>
</table>

### 5.3.4 Comparison with AES

- **Method of comparison**
  
  Speed measurement procedures similar to that described in paragraph "5.3.3. Speed evaluations" is performed on AES and compared with HyRAL. An accurate comparison could not be made however, as the measured AES uses source code that contains an assembler.

  Also note that comparisons with AES were carried out only for the coding process.

- **Method of evaluation**
  
  The following table shows the machines and OSs on which evaluations were carried out.

<table>
<thead>
<tr>
<th>Endian</th>
<th>CPU</th>
<th>Memory</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Little</td>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2GB</td>
<td>Windows XP Pro SP3</td>
</tr>
<tr>
<td>Little</td>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1GB</td>
<td>Windows XP Pro SP3</td>
</tr>
</tbody>
</table>

- 9 -
- Evaluation results

<<Coding process [128-bit]>>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1Block</th>
<th>Sec/1Block</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyRAL</td>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>676443936</td>
<td>668.277442000</td>
<td>0.000000284</td>
<td>1000000</td>
</tr>
<tr>
<td></td>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>659460428</td>
<td>656.071862000</td>
<td>0.000000356</td>
<td>1000000</td>
</tr>
<tr>
<td>AES</td>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>327552806</td>
<td>0.000000284</td>
<td>1000000</td>
<td>1000000</td>
</tr>
<tr>
<td></td>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>512242619</td>
<td>511.463520000</td>
<td>0.000000276</td>
<td>1000000</td>
</tr>
</tbody>
</table>

AES was approximately 51% faster than HyRAL when running a 128-bit coding process on an Intel CPU.
AES was approximately 22% faster than HyRAL when running a 128-bit coding process on an AMD CPU.

<<Coding process [256-bit]>>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CPU</th>
<th>Actual measurement in Hz</th>
<th>Clock</th>
<th>Clock/1Block</th>
<th>Sec/1Block</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyRAL</td>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>891542778</td>
<td>891.542778000</td>
<td>0.000000379</td>
<td>1000000</td>
</tr>
<tr>
<td></td>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>887742704</td>
<td>886.610207000</td>
<td>0.000000481</td>
<td>1000000</td>
</tr>
<tr>
<td>AES</td>
<td>Intel Core2 Duo 2.33GHz</td>
<td>2346708812</td>
<td>489961458</td>
<td>459.049990000</td>
<td>0.000000196</td>
<td>1000000</td>
</tr>
<tr>
<td></td>
<td>AMD Athlon64 2800+ 1.8GHz</td>
<td>1844040188</td>
<td>696486615</td>
<td>696.486615000</td>
<td>0.000000377</td>
<td>1000000</td>
</tr>
</tbody>
</table>

6. Evaluation of hardware integration
The following describe evaluations of an FPGA that was designed based on HyRAL FPGA design specifications.

6.1 Evaluation environment

6.1.1 FPGA
The target FPGA used was XC5VLX30 made by Xilinx.

6.1.2 Simulation environment
The simulator tool used was Model-Sim made by MentorGraphics.

6.1.3 Compilation and layout environment
The compilation and layout tool used was ISE made by Xilinx.

6.2 Evaluation results
Evaluations were made based on the logic verification performed on Model-Sim and the results of compilation and layout obtained from ISE.

6.2.1 Logic verification performed on Model-Sim
See 6.2.3 on cross checks between results of hardware verifications using Model-Sim and expected values obtained from software.
The simulation pattern used was provided by Laurel Bank Machines.
Verification was performed by coding 128 patterns of plain text for 10 patterns of keys for both single and double key modes.
The keys and plain text test data are the same as that used in the test vector.

6.2.2 Process time
The process time is the time required when processed at a clock speed of 20MHz (50ns).
- Subkey generation time
  BSTART-KEYEND(SinglaKayMode):6.3μs(126CLK)
  BSTART-KEYEND(DoubleKeyMode):12.55μs(251 CLK)
- Coded text generation time
  CSTART-EEND(SinglaKayMode):11.5μs(230CLK)
  CSTART-EEND(DoubleKayMode)16.0μs(320CLK)
- Decoded text generation time
  DSTART-DEND(SinglaKayMode)11.55μs(231CLK)
  DSTART-DEND(DoubleKayMode)16.05μs(321CLK)
- Subkey generation time: From BSTART to KEYEND.

BSTART-KEYGEN(SingleKeyMode): 6.3 μs

BSTART is the key generation start signal that is defined at the 0th bit at the address 0x0.
KEYEND is the signal that denotes the end of subkey generation.

BSTART-KEYGEN(DoubleKeyMode): 12.55 μs

- Coded text generation time: From CSTART to EEND.

CSTART-EEND(SingleKeyMode): 11.5 μs

CSTART-EEND(DoubleKeyMode): 16.0 μs

- Decoded text generation time: From DSTART to DEND.

DSTART-DEND(SingleKeyMode): 11.55 μs
6.2.3 Compilation and layout results were obtained using ISE made by Xilinx.

- Optimization = Speed

**HyRAL_ALL Project Status (01/01/2010–06:53:10)**

<table>
<thead>
<tr>
<th>Project File:</th>
<th>HyRAL_ALL_ise</th>
<th>Current State:</th>
<th>Placed and Routed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Name:</td>
<td>HyRAL128_main</td>
<td>Errors:</td>
<td>No Errors</td>
</tr>
<tr>
<td>Target Device:</td>
<td>xc5vlx30-3ff324</td>
<td>Warnings:</td>
<td>5 Warnings</td>
</tr>
<tr>
<td>ProductVersion:</td>
<td>ISE10.1–WebPACK</td>
<td>Routing Results:</td>
<td>All Signals Completely Routed</td>
</tr>
<tr>
<td>Design Goal:</td>
<td>Balanced</td>
<td>Timing Constraints:</td>
<td>All Constraints Met</td>
</tr>
<tr>
<td>Design Strategy:</td>
<td>Xilinx x Default (unlocked)</td>
<td>Final Timings Score:</td>
<td>0 (Timing Report)</td>
</tr>
</tbody>
</table>

**HyRAL_ALL Partition Summary**

No partition in formation was found.

**Device Utilization Summary**

- **Slice Logic Utilization**
  - Number of Slice Registers: 7,271 / 19,200 (37%)
  - Number used as Flip Flops: 7,271
  - Number of Slice LUTs: 9,211 / 19,200 (47%)
  - Number used as logic: 9,211 / 19,200 (47%)
  - Number using O6 output only: 9,211

- **Slice Logic Distribution**
  - Number of occupied Slices: 3,135 / 4,800 (65%)
  - Number of LUT Flip Flop pairs used: 11,025
  - Number with an unused Flip Flop: 3,754 / 11,025 (34%)
  - Number with an unused LUT: 1,814 / 11,025 (16%)
  - Number of fully used LUT–FF pairs: 5,457 / 11,025 (49%)
  - Number of unique control sets: 53

- **IO Utilization**
  - Number of bonded IOBs: 71 / 220 (32%)

- **Specific Feature Utilization**
  - Number of Block RAM/FIFO: 2 / 32 (6%)
  - Number using Block RAM only: 2

- **Total primitives used**
  - Number of 18k Block RAM used: 4

- **Total Memory used (KB)**
  - 72 / 1,152 (6%)

- **Number of BUF/BUFGCTRLs**
  - 2 / 32 (6%)

- **Number used as BUFGs**: 2

**Performance Summary**

- Final Timing Score: 0
- Pinout Data: Pinout report
- Routing Results: All Signals Completely Routed
- Clock Data: Clock Report
- Timing Constraints: All Constraints Met
Detailed Reports

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Status</th>
<th>Generated</th>
<th>Errors</th>
<th>Warnings</th>
<th>Infos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthesis Report</td>
<td>Current</td>
<td>±1 2 04:09:03 2010</td>
<td>0</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>Translation Report</td>
<td>Current</td>
<td>±1 2 06:35:41 2010</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Map Report</td>
<td>Current</td>
<td>±1 2 06:47:37 2010</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Place and Route Report</td>
<td>Current</td>
<td>±1 2 06:51:48 2010</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Static Timing Report</td>
<td>Current</td>
<td>±1 2 06:53:09 2010</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Bitgen Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Optimization = Area

HyRAL_ALL Project Status(01/01/2010–12:01:50)

<table>
<thead>
<tr>
<th>Project File:</th>
<th>HyRAL_ALL_ise</th>
<th>Current State:</th>
<th>Placed and Routed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Name:</td>
<td>HyRAL128_main</td>
<td>Errors:</td>
<td>No Errors</td>
</tr>
<tr>
<td>Target Device:</td>
<td>xc5vlx30-3ff324</td>
<td>Warnings:</td>
<td>751 Warnings</td>
</tr>
<tr>
<td>ProductVersion:</td>
<td>ISE10.1–WebPACK</td>
<td>Routing Results:</td>
<td>All Signals Completely Routed</td>
</tr>
<tr>
<td>Design Goal:</td>
<td>Area Reduction</td>
<td>Timing Constraints:</td>
<td>All Constraints Met</td>
</tr>
<tr>
<td>Design Strategy:</td>
<td>strategy1</td>
<td>Final Timings Score:</td>
<td>0(Timing Report)</td>
</tr>
</tbody>
</table>

HyRAL_ALL Partition Summary

No partition in formation was found.

Device Utilization Summary

<table>
<thead>
<tr>
<th>Slice Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
<th>Note(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Registers</td>
<td>7,239</td>
<td>19,200</td>
<td>37%</td>
<td></td>
</tr>
<tr>
<td>Number used as Flip Flops</td>
<td>7,239</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Slice LUTs</td>
<td>8,576</td>
<td>19,200</td>
<td>44%</td>
<td></td>
</tr>
<tr>
<td>Number used as logic</td>
<td>8,576</td>
<td>19,200</td>
<td>44%</td>
<td></td>
</tr>
<tr>
<td>Number using O6 output only</td>
<td>7,804</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number using O5 output only</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number using O5 end O6</td>
<td>766</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Slice Logic Distribution

| Number of occupied Slices | 3,135 | 4,800 | 65% |
| Number of LUT Flip Flop pairs used | 11,025 |       |
| Number with an unused Flip Flop | 3,754 | 11,025 | 34% |
| Number with an unused ULT | 1,814 | 11,025 | 16% |
| Number of fully used LUT–FF pairs | 5,457 | 11,025 | 49% |
| Number of unique control sets | 53 |       |

IO Utilization

| Number of bonded IOBs | 71 | 220 | 32% |

Specific Feature Utilization

| Number of Block RAM/FIFO | 2 | 32 | 6% |
| Number using Block RAM only | 2 |       |

Total primitives used

| Number of 18k Block RAM used | 4 |             |

Total Memory used(KB)

| 72 | 1,152 | 6% |

| Number of BUF/G/BUFGCTRLs | 2 | 32 | 6% |
| Number used as BUFGs | 2 |       |

Performance Summary

| Final Timing Score: | 0 | Pinout Data: | Pinout report |
| Routing Results: | All Signals Completely Routed | Clock Data: | Clock Report |
| Timing Constraints: | All Constraints Met |     |           |
6.2.4 Synthesis Report

The following is a verification of 4 warnings issued by ISE in its Synthesis Report.

```
WARNING:Xst2211 -"sbox_rom.v" line 108: Instantiating black box module 'sbox_rom'.
WARNING:Xst2211 -"sbox_rom.v" line 113: Instantiating black box module 'sbox_rom'.
WARNING:Xst2211 -"sbox_rom.v" line 118: Instantiating black box module 'sbox_rom'.
WARNING:Xst2211 -"sbox_rom.v" line 123: Instantiating black box module 'sbox_rom'.
```

These warnings cite that a module named sbox_rom, which was created at the CoreGenerator, is used in the black box. As such, these warnings pose no problems.

6.2.5 Translation Report

The following is a verification of 1 warning issued by ISE in its Translation Report.

```
WARNING:NgdBuild:478 –clock net RST_IBUF with clock driver RST_IBUF_BUFG drives no clock pins
```

The above warning cites that while an RST signal was generated by the DCM, it does not drive clock pins such as flip-flop operations. This is because an asynchronous reset is used in the circuitry, and this does not pose any problem.

6.2.6 Static Timing Report

We verified the contents of the Static Timing Report generated by ISE.

- **Optimization = Speed**

```
Timing constraint: TS_CLK=PERIODTIMEGRP "CLKI'20nsHIGH50%:
100886 paths analyzed, 27772 endpoints analyzed, 0 failing endpoints
0 timing errors detected. (0 setup errors, 0 hold errors)
Minimum period is 9.885ns.
```

The above shows that the worst operational frequency in the final configuration is 101MHz (period: 9.885ns).

- **Optimization = Area**

```
Timing constraint: TS_CLK=PERIODTIMEGRP "CLKI'20nsHIGH50%:
112569 paths analyzed, 25823 endpoints analyzed, 0 failing endpoints
0 timing errors detected. (0 setup errors, 0 hold errors)
Minimum periods 9.055ns.
```

The above shows that the worst operational frequency in the final configuration is 110MHz (period: 9.055ns).

6.3 Conclusion

By integrating HyRAL abbreviation and decoding into FPGA, and conducting a simulation to verify the operation of the logic circuitry as described in paragraph 6.2.1, we were able to verify the feasibility of abbreviation and decoding. Verifications for 1280 patterns of operations were conducted in both single and double key modes to verify the validity of the logic circuitry.

As for processing speed, the maximum speed was 572CLK (=251CLK [key generation] + 321 CLK [decoding]) as shown in paragraph 6.2.1.

The maximum frequency for compilation and layout was 110MHz, indicating that a time of approximately 5.179µs is required to complete all processes.

As for integration area, results show that there is almost no benefit from optimization. And while the
utilization rate was approximately 65% based on the number of slices used in Xilinx's XC5VLX30, this result reflects the availability of 128-, 192-, and 256-bit keys, and the fact that coding and decoding functions have also been included.

Additionally, the result also reflects the addition of a simple interface circuit to overcome the limitation in the number of IO pins available in FPGA.

The interface circuitry performed a minimum of approximately 1,000 flip-flop operations and used an accompanying logic circuitry. Approximately 6,200 flip-flop operations were performed for coding and decoding.

7. Side channel attacks

7.1 Protection against cache attacks

Similar to AES, HyRAL makes extensive use of table lookups. Therefore, considerable cost must be expended to protect against cache attacks.

The most direct method of attacking cache would be to analyze the subkeys one by one by gathering information on how the targeted coding program uses cache memory so as to directly control the cache memory. This is done using a program that monitors the same processor on which the coding program is running.

All coding methods that use large table lookups, not only HyRAL, are vulnerable to this type of attack. To this end, while there is a preference for nonlinear processes that do not involve table lookups, neither of these methodologies can be said to be superior over the other from the standpoint that it would be difficult for nonlinear methods to deliver comparable benefits, and based on comparisons with regard to other modes of attack.

While HyRAL uses table lookup, we also consider measures to protect against cache attacks. One method we are considering is to use 1 table as opposed to 4 so as to prevent the distribution of access to tables. This method however, will result in a considerable reduction in efficiency.

Another method involves the use of buzz processes, which will be run for the first several rounds. Buzz processes have no utility and are again a cause for a reduction in efficiency.

As for threats other than cache attacks or other forms of attack that target the algorithm, we will leave the option available on whether or not to use methods equipped with countermeasures depending on security levels.

We also note the following points as they may become important going forward, although they may not be relevant to the current case.

Traditionally, it has been presupposed that all coding modes using symmetric-key coding use static keys. CBC, CFB, and OFB are all based on ECB. As such, attackers have an advantage in that they are able to mount attacks on keys that are constantly in the same state. Cache attacks are a form of this type of attack. If keys are changed with the passage of time, statistical methods used in traditional methods of attack are rendered ineffective. While the idea of variable keys has been around since the announcement of DES, this has not been adopted as a mode of coding because of its lack of affinity with hardware. Meanwhile, from a real-world application perspective, little else than key coding operations rely on ECB alone, and there is considerable need for confidentiality and integrity for a range of data including communication data and files.

Currently, a large number of proposals for hash functions are being made, and recently, the use of internal
information chain methods has been on the rise. This is referred to in the report as "test information" or "change value."
We believe that cache attacks will be rendered ineffective if this method can be used in block coding.
As the next step in the development of HyRAL, we will incorporate a design that takes into account internal information chains, and plan to expand its scope of applications from the traditional static use of keys to their dynamic use.
The ECB mode, which uses keys statically, is similar to master key-based key coding operations, and because master keys are presumably sealed in an anti-manipulation device, we believe this should not be kept in environments that can be exposed to cache attacks.
Based on this line of thought, we will continue to study and announce measures that prioritize safety, based on the assumption that the ECB mode will be used only in a limited number of scenarios, and with the understanding that these measures must not have a significant impact on overall system performance even though their overhead with regard to cache attacks may be large.

7.2 Protection against SPA and DPA attacks
As for attacks where measurements for hardware power consumption and voltage variations are used to obtain the values of secret keys, the code will be installed in such a way that these variables are smoothed out to reduce the relevance of hardware measurements as much as possible. We plan to incorporate measures such as those cited in the guidelines. The following is one possible idea for a physical countermeasure as well as a measure for incorporated software.
These attacks are based on information (key and data), particularly with regard to hamming weight. If this hamming weight can be smoothed, we believe that this mode of attack will be rendered useless for the attacker.
Two logic processes must be embedded: one to perform the correct calculation and the other to perform dummy calculations to achieve smoothness.
Most methods generally obtain the XOR of the secret key and data, and search the table. We will create a scheme where the input data passes through two processes.
One process is the correct one where XOR is obtained for the secret key, and the other obtains the NOT value and XOR of the secret key, with data passing through in similar fashion. While this will result in a doubling of power consumption, it will provide protection against attacks based on measurements of electrical characteristics. Consideration will be needed as to whether or not to implement this across the entire algorithm.
Anti-manipulation devices are devices that can neither be opened nor allow direct measurements of the chip it contains.
It is common practice to design the device so that the secret key is erased if the device is opened.
Logic-level countermeasures become necessary for devices such as smart cards however, because they require a supply of power from an external device.