The RC6 Block Cipher: A simple fast secure AES proposal

Ronald L. Rivest  MIT
Matt Robshaw    RSA Labs
Ray Sidney      RSA Labs
Yiqun Lisa Yin RSA Labs

(August 21, 1998)
Outline

- Design Philosophy
- Description of RC6
- Implementation Results
- Security
- Conclusion
Design Philosophy

- Leverage our experience with RC5: use data-dependent rotations to achieve a high level of security.
- Adapt RC5 to meet AES requirements
- Take advantage of a new primitive for increased security and efficiency: 32x32 multiplication, which executes quickly on modern processors, to compute rotation amounts.
Description of RC6
Description of RC6

- **RC6-w/r/b parameters:**
  - Word size in bits: \( w = 32 \) \( (\lg(w) = 5) \)
  - Number of rounds: \( r = 20 \)
  - Number of key bytes: \( b = 16, 24, \text{ or } 32 \)

- **Key Expansion:**
  - Produces array \( S[0 \ldots 2r + 3] \) of \( w \)-bit round keys.

- **Encryption and Decryption:**
  - Input/Output in 32-bit registers \( A, B, C, D \)
<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A + B$</td>
<td>Addition modulo $2^w$</td>
</tr>
<tr>
<td>$A - B$</td>
<td>Subtraction modulo $2^w$</td>
</tr>
<tr>
<td>$A \oplus B$</td>
<td>Exclusive-Or</td>
</tr>
<tr>
<td>$A \ll B$</td>
<td>Rotate $A$ left by amount in low-order $\lg(w)$ bits of $B$</td>
</tr>
<tr>
<td>$A \gg B$</td>
<td>Rotate $A$ right, similarly</td>
</tr>
<tr>
<td>$(A,B,C,D) = (B,C,D,A)$</td>
<td>Parallel assignment</td>
</tr>
<tr>
<td>$A \times B$</td>
<td>Multiplication modulo $2^w$</td>
</tr>
</tbody>
</table>
RC6 Encryption (Generic)

\[ B = B + S[0] \]
\[ D = D + S[1] \]

for i = 1 to r do
\{
\[ t = (B \times (2B + 1)) \ll \lg(w) \]
\[ u = (D \times (2D + 1)) \ll \lg(w) \]
\[ A = ((A \oplus t) \ll u) + S[2i] \]
\[ C = ((C \oplus u) \ll t) + S[2i + 1] \]
\[ (A, B, C, D) = (B, C, D, A) \]
\}
\[ A = A + S[2r + 2] \]
\[ C = C + S[2r + 3] \]
RC6 Encryption (for AES)

\[
B = B + S[0]
\]
\[
D = D + S[1]
\]
for i = 1 to 20 do
\{
  \[
  t = (B \times (2B + 1)) \ll 5
  \]
  \[
  u = (D \times (2D + 1)) \ll 5
  \]
  \[
  A = ((A \oplus t) \ll u) + S[2i]
  \]
  \[
  C = ((C \oplus u) \ll t) + S[2i + 1]
  \]
  (A, B, C, D) = (B, C, D, A)
\}
\[
A = A + S[42]
\]
\[
C = C + S[43]
\]
RC6 Decryption (for AES)

\[
C = C - S[43]
\]
\[
A = A - S[42]
\]
for \(i = 20\) downto 1 do
\{
\[
(A, B, C, D) = (D, A, B, C)
\]
\[
u = (D \times (2D + 1)) \ll 5
\]
\[
t = (B \times (2B + 1)) \ll 5
\]
\[
C = (((C - S[2i + 1]) \gg t) \oplus u
\]
\[
A = (((A - S[2i]) \gg u) \oplus t
\]
\}
\[
D = D - S[1]
\]
\[
B = B - S[0]
\]
Key Expansion (Same as RC5’s)

- Input: array L[0 ... c-1] of input key words
- Output: array S[0 ... 43] of round key words
- Procedure:
  - S[0] = 0xB7E15163
  - for i = 1 to 43 do S[i] = S[i-1] + 0x9E3779B9
  - A = B = i = j = 0
  - for s = 1 to 132 do
    - { A = S[i] = (S[i] + A + B) <<< 3
    - B = L[j] = (L[j] + A + B) <<< (A + B)
    - i = (i + 1) mod 44
    - j = (j + 1) mod c
    - }

From RC5 to RC6
in seven easy steps
(1) Start with RC5

RC5 encryption inner loop:

```
for i = 1 to r do
{
    A = (((A ⊕ B) <<< B) + S[i])
    (A, B) = (B, A)
}
```

Can RC5 be strengthened by having rotation amounts depend on all the bits of B?
Better rotation amounts?

- **Modulo function?**
  Use low-order bits of \((B \mod d)\)
  Too slow!

- **Linear function?**
  Use high-order bits of \((c \times B)\)
  Hard to pick \(c\) well!

- **Quadratic function?**
  Use high-order bits of \((B \times (2B+1))\)
  Just right!
**B x (2B+1) is one-to-one mod 2^w**

*Proof:* By contradiction. If \( B \neq C \) but
\[
B \times (2B + 1) = C \times (2C + 1) \pmod{2^w}
\]
then
\[
(B - C) \times (2B+2C+1) = 0 \pmod{2^w}
\]
But \((B-C)\) is nonzero and \((2B+2C+1)\) is odd; their product can't be zero! □

*Corollary:*
\( B \) uniform \( \rightarrow \) \( B \times (2B+1) \) uniform
(and high-order bits are uniform too!)
The high-order bits of $B \times (2B + 1)$ depend on all the bits of $B$.

Let $B = B_{31}B_{30}B_{29} \ldots B_1B_0$ in binary.

Flipping bit $i$ of input $B$
- Leaves bits $0 \ldots i-1$ of $f(B)$ unchanged,
- Flips bit $i$ of $f(B)$ with probability one,
- Flips bit $j$ of $f(B)$, for $j > i$, with probability approximately $1/2$ ($1/4 \ldots 1$),
- is likely to change some high-order bit.
(2) Quadratic Rotation Amounts

for $i = 1$ to $r$ do

{$$
t = (B \times (2B + 1)) \ll 5
$$

$$A = ((A \oplus B) \ll t) + S[i]
$$

$(A, B) = (B, A)$

}
(3) *Use \( t \), not \( B \), as xor input*

```plaintext
for i = 1 to r do
{
    \[ t = (B \times (2B + 1)) \ll 5 \]
    \[ A = ((A \oplus t) \ll t) + S[i] \]
    \[ (A, B) = (B, A) \]
}
```

Now AES requires 128-bit blocks. We could use two 64-bit registers, but 64-bit operations are poorly supported with typical C compilers...
(4) Do two RC5’s in parallel

Use four 32-bit regs \((A,B,C,D)\), and do RC5 on \((C,D)\) in parallel with RC5 on \((A,B)\):

\[
\text{for } i = 1 \text{ to } r \text{ do}
\{
\begin{align*}
    t &= (B \times (2B + 1)) \ll 5 \\
    A &= ((A \oplus t) \ll t) + S[2i] \\
    (A, B) &= (B, A) \\

    u &= (D \times (2D + 1)) \ll 5 \\
    C &= ((C \oplus u) \ll u) + S[2i + 1] \\
    (C, D) &= (D, C)
\end{align*}
\}
\]
(5) Mix up data between copies

Switch rotation amounts between copies, and cyclically permute registers instead of swapping:

for i = 1 to r do
    {
        \( t = (B \times (2B + 1)) \ll 5 \)
        \( u = (D \times (2D + 1)) \ll 5 \)
        \( A = ((A \oplus t) \ll u) + S[2i] \)
        \( C = ((C \oplus u) \ll t) + S[2i + 1] \)
        \( (A, B, C, D) = (B, C, D, A) \)
    
}
One Round of RC6
(6) Add Pre- and Post-Whitening

\[
\begin{align*}
B &= B + S[0] \\
D &= D + S[1] \\
\text{for } i &= 1 \text{ to } r \text{ do} \\
\{ & \quad t = (B \times (2B + 1)) \ll 5 \\
& \quad u = (D \times (2D + 1)) \ll 5 \\
& \quad A = ((A \oplus t) \ll u) + S[2i] \\
& \quad C = ((C \oplus u) \ll t) + S[2i + 1] \\
(A, B, C, D) &= (B, C, D, A) \\
\}
\end{align*}
\]

\[
\begin{align*}
A &= A + S[2r + 2] \\
C &= C + S[2r + 3]
\end{align*}
\]
(7) Set \( r = 20 \) for high security

\[
B = B + S[0] \\
D = D + S[1] \\
\text{for } i = 1 \text{ to } 20 \text{ do} \\
\{
\begin{align*}
\begin{align*}
\top &= (B \times (2B + 1)) \ll 5 \\
u &= (D \times (2D + 1)) \ll 5 \\
A &= ((A \oplus \top) \ll u) + S[2i] \\
C &= ((C \oplus u) \ll \top) + S[2i + 1] \\
(A, B, C, D) &= (B, C, D, A)
\end{align*}
\end{align*}
\}
\]

\[
A = A + S[42] \\
C = C + S[43]
\]

Final RC6
RC6 Implementation Results
## CPU Cycles / Operation

<table>
<thead>
<tr>
<th>Operation</th>
<th>Java</th>
<th>Borland C</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td>110000</td>
<td>2300</td>
<td>~1000</td>
</tr>
<tr>
<td>Encrypt</td>
<td>16200</td>
<td>616</td>
<td>254</td>
</tr>
<tr>
<td>Decrypt</td>
<td>16500</td>
<td>566</td>
<td>254</td>
</tr>
</tbody>
</table>

Less than two clocks per bit of plaintext!
## Operations/Second (200MHz)

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>Borland C</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup</strong></td>
<td>1820</td>
<td>86956</td>
<td>~200000</td>
</tr>
<tr>
<td><strong>Encrypt</strong></td>
<td>12300</td>
<td>325000</td>
<td>787000</td>
</tr>
<tr>
<td><strong>Decrypt</strong></td>
<td>12100</td>
<td>353000</td>
<td>788000</td>
</tr>
</tbody>
</table>
Encryption Rate (200MHz)

<table>
<thead>
<tr>
<th></th>
<th>MegaBytes / second</th>
<th>MegaBits / second</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Java</td>
<td>Borland C</td>
</tr>
<tr>
<td>Encrypt</td>
<td>0.197</td>
<td>5.19</td>
</tr>
<tr>
<td></td>
<td>1.57</td>
<td>41.5</td>
</tr>
<tr>
<td>Decrypt</td>
<td>0.194</td>
<td>5.65</td>
</tr>
<tr>
<td></td>
<td>1.55</td>
<td>45.2</td>
</tr>
</tbody>
</table>

Over 100 Megabits / second!
On an 8-bit processor

- On an Intel MCS51 (1 Mhz clock)
- Encrypt/decrypt at 9.2 Kbits/second (13535 cycles/block; from actual implementation)
- Key setup in 27 milliseconds
- Only 176 bytes needed for table of round keys.
- Fits on smart card (< 256 bytes RAM).
Custom RC6 IC

- 0.25 micron CMOS process
- One round/clock at 200 MHz
- Conventional multiplier designs
- $0.05 \text{ mm}^2$ of silicon
- 21 milliwatts of power
- Encrypt/decrypt at 1.3 Gbits/second
- With pipelining, can go faster, at cost of more area and power
RC6 Security Analysis
Analysis procedures

- Intensive analysis, based on most effective known attacks (e.g. linear and differential cryptanalysis)
- Analyze not only RC6, but also several “simplified” forms (e.g. with no quadratic function, no fixed rotation by 5 bits, etc...)}
Linear analysis

- Find approximations for \( r-2 \) rounds.
- Two ways to approximate \( A = B \ll C \):
  - with one bit each of \( A, B, C \) (type I)
  - with one bit each of \( A, B \) only (type II)
  - each have bias \( 1/64 \); type I more useful
- Non-zero bias across \( f(B) \) only when input bit = output bit. (Best for lsb.)
- Also include effects of multiple linear approximations and linear hulls.
Security against linear attacks

Estimate of number of plaintext/ciphertext pairs required to mount a linear attack.

(Only $2^{128}$ such pairs are available.)

<table>
<thead>
<tr>
<th>Rounds</th>
<th>Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>$2^{47}$</td>
</tr>
<tr>
<td>12</td>
<td>$2^{83}$</td>
</tr>
<tr>
<td>16</td>
<td>$2^{119}$</td>
</tr>
<tr>
<td>20</td>
<td>$2^{155}$ Infeasible</td>
</tr>
<tr>
<td>24</td>
<td>$2^{191}$</td>
</tr>
</tbody>
</table>
Differential analysis

- Considers use of (iterative and non-iterative) (r-2)-round differentials as well as (r-2)-round characteristics.
- Considers two notions of “difference”:
  - exclusive-or
  - subtraction (better!)
- Combination of quadratic function and fixed rotation by 5 bits very good at thwarting differential attacks.
An iterative RC6 differential

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1&lt;&lt;16</td>
<td>1&lt;&lt;11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1&lt;&lt;11</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1&lt;&lt;s</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1&lt;&lt;26</td>
<td>1&lt;&lt;s</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1&lt;&lt;26</td>
<td>1&lt;&lt;21</td>
<td>0</td>
<td>1&lt;&lt;v</td>
</tr>
<tr>
<td></td>
<td>1&lt;&lt;21</td>
<td>1&lt;&lt;16</td>
<td>1&lt;&lt;v</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1&lt;&lt;16</td>
<td>1&lt;&lt;11</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Probability** = $2^{-91}$
Security against differential attacks

Estimate of number of plaintext pairs required to mount a differential attack.

(Only \(2^{128}\) such pairs are available.)

<table>
<thead>
<tr>
<th>Rounds</th>
<th>Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>(2^{56})</td>
</tr>
<tr>
<td>12</td>
<td>(2^{97})</td>
</tr>
<tr>
<td>16</td>
<td>(2^{190}) <strong>Infeasible</strong></td>
</tr>
<tr>
<td>20</td>
<td>(2^{238})</td>
</tr>
<tr>
<td>24</td>
<td>(2^{299})</td>
</tr>
</tbody>
</table>
Security of Key Expansion

- Key expansion is identical to that of RC5; no known weaknesses.
- No known weak keys.
- No known related-key attacks.
- Round keys appear to be a “random” function of the supplied key.
- Bonus: key expansion is quite “one-way”---difficult to infer supplied key from round keys.
Conclusion

RC6 more than meets the requirements for the AES; it is
- simple,
- fast, and
- secure.

For more information, including copy of these slides, copy of RC6 description, and security analysis, see www.rsa.com/rsalabs/aes
(The End)