Algoritma Kriptografi Kunci Simetris "Camellia"

Tugas Akhir
Keamanan Sistem Informasi
(EC - 5010)

oleh:
Ahmad Rifqi Hadiyanto (13200013)

Institut Teknologi Informasi
Departemen Teknik Elektro
Institut Teknologi Bandung
2004

Daftar Isi

Algoritma Kunci Simetris "Camellia"

1. Pendahuluan

Dewasa ini, dalam dunia dengan arus informasi yang semakin global, kriptografi telah menjadi suatu bagian yang tidak dapat dipisahkan dari sistem keamanan jaringan. Ada berbagai algoritma kriptografi yang sekarang ini telah dan sedang dikembangkan, baik algoritma kunci simetris ataupun asimetris (pembagian berdasarkan kunci). Isu yang seringkali berkaitan dengan kriptografi adalah tingkat keamanan algoritma dan kecepatan transfer data yang mampu diberikan oleh implementasi algoritma tersebut, walaupun masih ada beberapa parameter lain, seperti: kemampuan sebuah algoritma untuk diimplementasikan dalam berbagai macam platform, kebutuhan resource yang kecil (*memori* pada "programmable device", atau *area* pada "custom device"),dll.
Camellia adalah sebuah algoritma kriptografi kunci simetris yang bekerja pada ukuran blok 128 bit dengan panjang kunci 128-bit, 192-bit, atau 256-bit. Camellia pertama kali dikembangkan secara bersama oleh NTT dan Mitsubishi Electric Corporation pada tahun 2000 . Kedua instansi ini pernah mengembangkan algoritma kriptografi yang cukup terkenal: E2 (dikembangkan oleh NTT) dan MISTY (dikembangkan oleh Mitsubishi) sehingga diharapkan *Camellia* mengadopsi beberapa fitur menguntungkan dari kedua macam algoritma kriptogafi tersebut (lihat referensi [4]).

Tulisan ini akan membahas algoritma kriptografi kunci simetrik "Camellia" mulai dari bab 1 yaitu pendahuluan, bab 2 tentang notasi dan konvensi yang digunakan dalam dokumen ini, bab 2 menjelaskan struktur algoritma Camellia, bab 3 tentang Komponen penyusun Camellia, serta beberapa hal dasar dalam mengimplementasikannya pada piranti FPGA (Field Programmable Gate Array) yang akan dijelaskan pada bab 5.

2. Notasi dan konvensi

2.1 Radix

Dalam tulisan ini akan digunakan awalan *0x* untuk menunjukkan bahwa suatu bilangan adalah bilangan heksadesimal.

2.2 Notasi

Dalam tulisan ini digunakan beberapa notasi sebagai berikut:.

- B menunjukkan sebuah ruang vektor dengan 8-bit (1 byte) elemen; dimana $B = GF(2)^8$.
- W menunjukkan sebuah ruang vektor dengan 32-bit (word) elemen; dimana $W = B^4$.
- L menunjukkan sebuah ruang vektor dengan 64-bit (double word) elemen; dimana $L = B^8$.
- Q menunjukkan sebuah ruang vektor dengan 128-bit (quad word) elemen; dimana $Q = B^{16}$.
- Sebuah elemen dengan akhiran $_{(n)}$ (misal: $x_{(n)}$) menunjukkan bahwa elemen tersebut memiliki panjang n-bit.
- Sebuah elemen dengan akhiran $_L$ (misal: x_L) menunjukkan setengah bagian sebelah kiri dari x.
- Sebuah elemen dengan akhiran $_R$ (e.g. x_R) menunjukkan setengah bagian sebelah kanan dari x.

2.3 Keterangan simbol

$\oplus$	operasi bitwise exclusive-OR.
$\|\|$	hubungan serangkai antara dua buah operand.
$<<<n$	rotasi "left circular"dari sebuah operand sepanjang n bit.
$\cap$	The bitwise AND operation.
$\cup$	operasi bitwise OR.
$\overline{x}$	komplemen bitwise dari x.

2.4 Tata urutan Bit/Byte

Tulisan ini menggunakan tata urutan "big endian". Contoh berikut menunjukkan secara berurutan bagaimana cara menyusun sebuah nilai 128-bit $Q_{(128)}$ dari dua buah nilai 64-bit $L_{i(64)}(i = 1,2)$, empat buah nilai 32-bit $W_{i(32)}(i = 1,2,3,4)$, enam belas nilai 8-bit $B_{i(8)}(i = 1,2,...,16)$, atau dari 128 buah nilai 1-bit $E_{i(1)}(i = 1,2,...,128)$.

$$
\begin{aligned}
Q_{(128)} &= L_{1(64)} \| L_{2(64)} \\
&= W_{1(32)} \| W_{2(32)} \| W_{3(32)} \| W_{4(32)} \\
&= B_{1(8)} \| B_{2(8)} \| B_{3(8)} \| B_{4(8)} \| \ldots\ldots \| B_{13(8)} \| B_{14(8)} \| B_{15(8)} \| B_{16(8)} \\
&= E_{1(1)} \| E_{2(1)} \| E_{3(1)} \| E_{4(1)} \| \ldots\ldots\ldots\ldots\ldots\ldots \| E_{125(1)} \| E_{126(1)} \| E_{127(1)} \| E_{128(1)}
\end{aligned}
$$

Contoh Numerik:

$Q_{(128)} = 0x0123456789ABCDEF0011223344556677_{(128)}$

$L_{1(64)} = Q_{L(64)} = 0x0123456789ABCDEF_{(64)}$

$L_{2(64)} = Q_{R(64)} = 0x0011223344556677_{(64)}$

$W_{1(32)} = L_{1L(32)} = 0x01234567_{(32)}$

$W_{2(32)} = L_{1R(32)} = 0x89ABCDEF_{(32)}$

$W_{3(32)} = L_{2L(32)} = 0x00112233_{(32)}$

$W_{4(32)} = L_{2R(32)} = 0x44556677_{(32)}$

$B_{1(8)} = 0x01_{(8)}$, $B_{2(8)} = 0x23_{(8)}$, $B_{3(8)} = 0x45_{(8)}$, $B_{4(8)} = 0x67_{(8)}$,

$B_{5(8)} = 0x89_{(8)}$, $B_{6(8)} = 0xAB_{(8)}$, $B_{7(8)} = 0xCD_{(8)}$, $B_{8(8)} = 0xEF_{(8)}$,

$B_{9(8)} = 0x00_{(8)}$, $B_{10(8)} = 0x11_{(8)}$, $B_{11(8)} = 0x22_{(8)}$, $B_{12(8)} = 0x33_{(8)}$,

$B_{13(8)} = 0x44_{(8)}$, $B_{14(8)} = 0x55_{(8)}$, $B_{15(8)} = 0x66_{(8)}$, $B_{16(8)} = 0x77_{(8)}$,

$E_{1(1)} = 0_{(1)}$, $E_{2(1)} = 0_{(1)}$, $E_{3(1)} = 0_{(1)}$, $E_{4(1)} = 0_{(1)}$,

$E_{5(1)} = 0_{(1)}$, $E_{6(1)} = 0_{(1)}$, $E_{7(1)} = 0_{(1)}$, $E_{8(1)} = 1_{(1)}$,

.
.
.

$E_{121(1)} = 0_{(1)}$, $E_{122(1)} = 1_{(1)}$, $E_{123(1)} = 1_{(1)}$, $E_{124(1)} = 1_{(1)}$,

$E_{125(1)} = 0_{(1)}$, $E_{126(1)} = 1_{(1)}$, $E_{127(1)} = 1_{(1)}$, $E_{128(1)} = 1_{(1)}$

$$Q_{(128)} <<<_1 = E_{2(1)} \| E_{3(1)} \| E_{4(1)} \| E_{5(1)} \| \ldots\ldots \| E_{125(1)} \| E_{126(1)} \| E_{127(1)} \| E_{128(1)} \| E_{1(1)}$$
$$= 0x02468ACF13579BDE0022446688AACCEE_{(128)}$$

3. Struktur Camellia

3.1 Keterangan fungsi dan variabel

$M_{(128)}$	Blok plainteks
$C_{(128)}$	Blok chiperteks
K	Kunci rahasia, yang panjangnya 128, 192, atau 256 bit.
$kw_{t(64)}, k_{u(64)}, kl_{v(64)}$	subkunci (t =1,2,3,4) (u =1,2,...,18) (v =1,2,3,4) untuk kunci rahasia 128 bit. (t =1,2,3,4) (u =1,2,...,24) (v =1,2,...,6) untuk kunci rahasia 192 dan 256 bit.
$Y_{(64)}= F(X_{(64)}, k_{(64)})$	Fungsi F yang mentransformasi input 64bit, $X_{(64)}$ menghasilkan output 64 bit, Y(64) menggunakan subkey 64 bit, $k_{(64)}$.
$Y_{(64)}= FL(X_{((64)},k_{(64)})$	Fungsi FL yang mentransformasi input 64bit menghasilkan output 64 bit menggunakan subkey 64 bit.
$Y_{(64)}= FL^{-1}(X_{((64)},k_{(64)})$	Fungsi FL^{-1} yang mentransformasi input 64bit menghasilkan output 64 bit menggunakan subkey 64 bit.
$Y_{(64)}= S(X_{(64)})$	Fungsi S yang mentransformasi input 64bit menghasilkan output 64 bit.
$Y_{(64)}= P(X_{(64)})$	Fungsi P yang mentransformasi input 64bit menghasilkan output 64.
$y_{(8)}= si(X_{(8)})$	s-box yang mentransformasi input 8 bit menghasilkan output 8 bit (i =1,2,3,4).

3.2. Prosedur Enkripsi

3.2.1 kunci 128 bit

Gambar1 menunjukkan prosedur enkripsi untuk kunci 128 bit. Bagian pengacakan data memiliki struktur 18 tahap (round) feistel dengan 2 lapisan (layer) fungsi FL/FL^{-1} setelah tahap ke-6 dan tahap ke-12, dan operasi XOR 128 bit sebelum tahap pertama dan setelah tahap terakhir. Bagian penjadwalan kunci membangkitkan subkunci $kw_{t(64)}$(t =1,2,3,4), $k_{u(64)}$(u =1,2,...,18) dan $kl_{v(64)}$(v =1,2,3,4) dari kunci rahasia K; lihat seksi 3.4 untuk informasi detail dari bagian penjadwalan kunci.
Pada bagian pengacakan data, plainteks pertama $M_{(128)}$ di XOR dengan $kw_{1(64)}||kw_{2(64)}$ dan dipecah menjadi $L_{0(64)}$ dan $R_{0(64)}$ yang panjangnya sama, sehingga dapat ditulis sebagai berikut: $M_{(128)} \oplus (kw_{1(64)}||kw_{2(64)}) = L_{0(64)}||R_{0(64)}$.
Kemudian, operasi berikut akan dilakukan dari r = 1 sampai r = 18, kecuali untuk r = 6 dan 12:

$$L_r = R_{r-1} \oplus F(L_{r-1}, k_r),$$
$$R_r = L_{r-1}.$$

Untuk r = 6 dan 12, operasi berikut ini dilakukan:

$$L'r = R_{r-1} \oplus F(L_{r-1}, k_r),$$
$$R_r = L_{r-1},$$

$$Lr = FL(L'r, kl_{2r/6-1}),$$
$$Rr = FL^{-1}(R'_r, kl_{2r/6}).$$

Terakhir, $R_{18(64)}$ dan $L_{18(64)}$ dirangkai dan diXORed dengan $kw_{3(64)}||kw_{4(64)}$. Nilai resultannya adalah chiperteks yang dapat ditulis:

$$C_{(128)}=(R_{18(64)}||L_{18(64)}) \oplus (kw_{3(64)}||kw_{4(64)}).$$

3.2.2 Kunci 192 bit dan 256 bit

Gambar 2 menunjukkan prosedur enkripsi untuk kunci 192 bit atau 256 bit. Bagian pengacakan data memiliki struktur 24 tahap feistel dengan 3 buah lapisan fungsi FL/FL^{-1} setelah tahap ke-6, ke-12, dan ke-18, dan operasi XOR 128 bit sebelum tahap pertama dan setelah tahap terakhir. Bagian penjadwal kunci membangkitkan subkunci $kw_{t(64)}$(t =1,2,3,4), $k_{u(64)}$(u =1,2,...,24), dan $kl_{v(64)}$(v =1,2,...,6) dari kunci rahasia K.

Pada bagian pengacakan data, pertama-tama chiperteks $C_{(128)}$ di-XOR dengan $kw_{3(64)}||kw_{4(64)}$ dan dibagi menjadi $R_{18(64)}$ and $L_{18(64)}$ dengan panjang yang sama, sehingga dapat ditulis: $M_{(128)} \oplus (kw_{1(64)}||kw_{2(64)}) = L_{0(64)}||R_{0(64)}$. Kemudian, lakukan operasi berikut dari r = 1 sampai 24, kecuali untuk r = 6, 12, dan 18:

$$Lr = Rr-1 \oplus F(Lr-1, kr),$$
$$Rr = Lr-1.$$

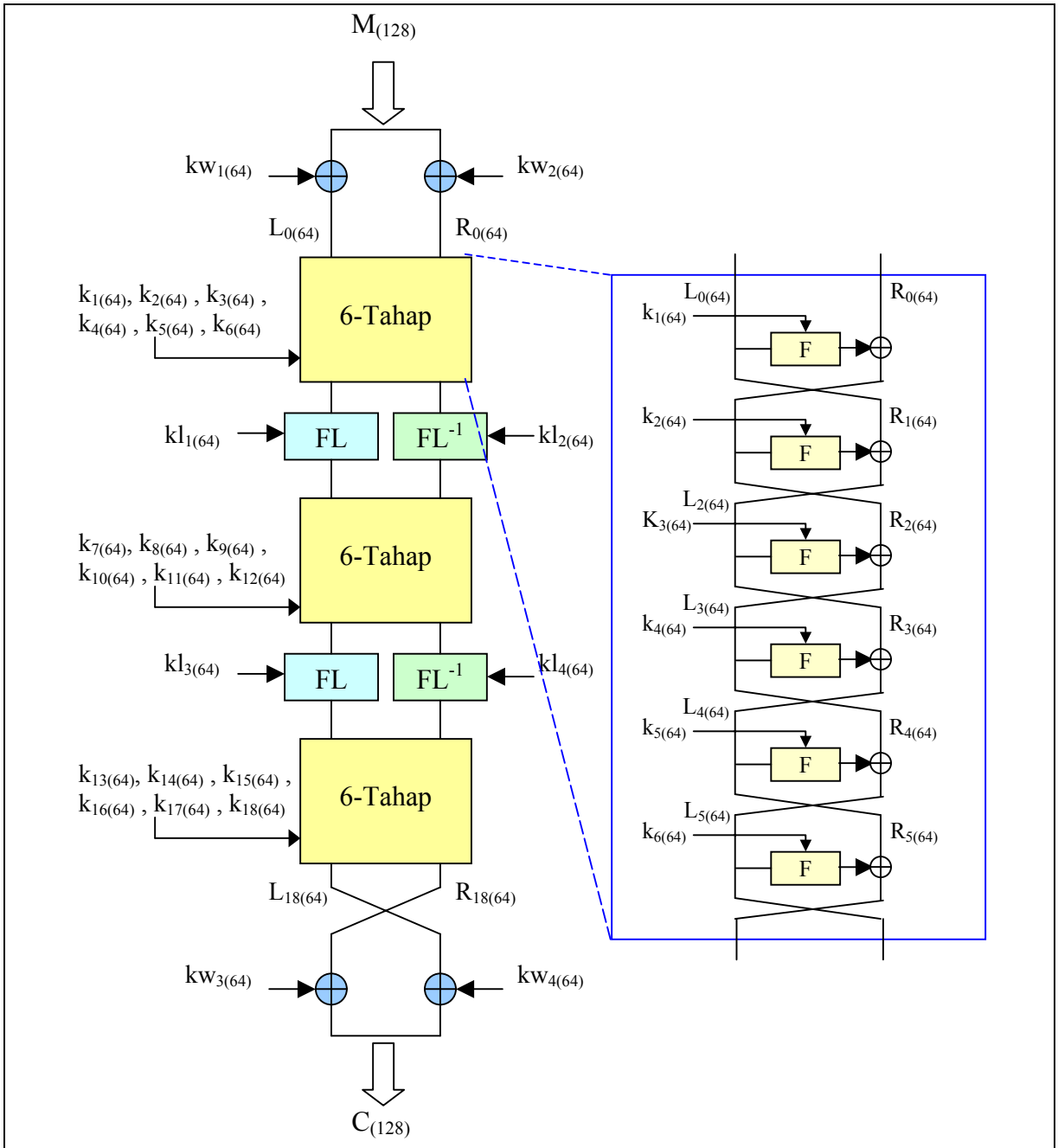
Untuk r = 6, 12, dan 18, lakukan operasi berikut:

$$L'_r = R_{r-1} \oplus F(L_{r-1}, k_r),$$
$$R_r = L_{r-1},$$
$$L_r = FL(L'r, kl_{2r/6-1}),$$
$$R_r = FL^{-1}(R'_r, kl_{2r/6}).$$

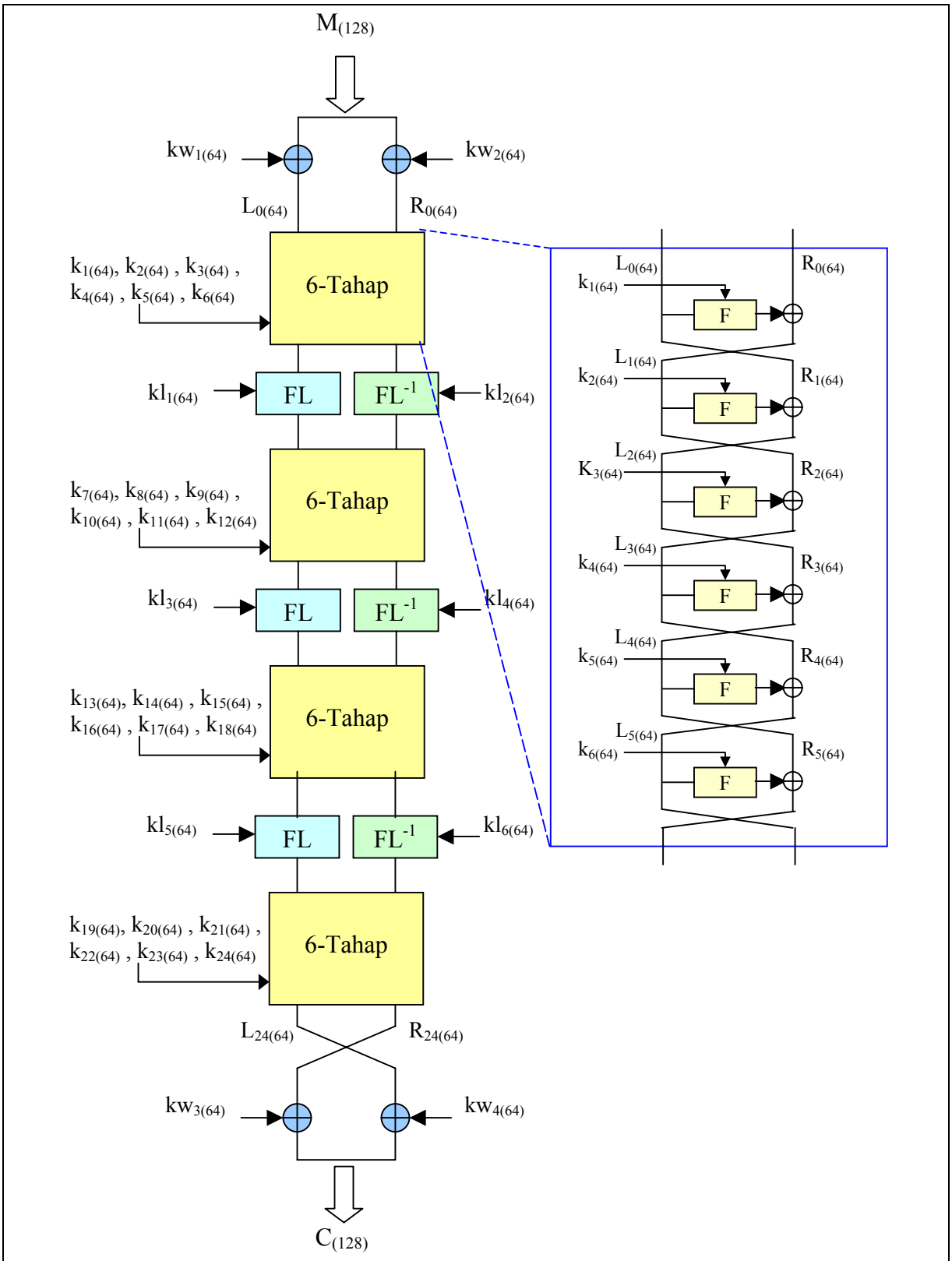
Terakhir, $R_{24(64)}$ dan $L_{24(64)}$ dirangkai dan di-XOR dengan $kw_{3(64)}||kw_{4(64)}$. Nilai resultannya adalah chiperteks yang dapat ditulis:

$$C_{(128)}=(R_{18(64)}||L_{18(64)}) \oplus (kw_{3(64)}||kw_{4(64)}).$$

Lihat seksi 4 untuk detail dari fungsi F dan FL/FL^{-1}.



Gambar 1. Prosedur Enkripsi Camellia untuk panjang kunci 128 bit



Gambar 2. Prosedur Enkripsi Camellia untuk panjang kunci 192 dan 256 bit.

3.3 Prosedur Dekripsi

3.3.1 Kunci 128 bit

Prosedur dekripsi dari Camellia dilakukan dengan jalan yang sama dengan prosedur enkripsi dengan membalik urutan dari subkunci.

Gambar 3 menunjukkan prosedur dekripsi untuk kunci 128 bit. Bagian pengacakan data memiliki struktur 18 tahap (round) feistel dengan 2 lapisan (layer) fungsi FL/FL^{-1} setelah tahap ke-6 dan tahap ke-12, dan operasi XOR 128 bit sebelum tahap pertama dan setelah tahap terakhir. Bagian penjadwal kunci membangkitkan subkunci $kw_{t(64)}(t=1,2,3,4)$, $k_{u(64)}(u=1,2,...,18)$, dan $kl_{v(64)}(v=1,2,3,4)$ dari kunci rahasia K. Lihat seksi 3.4 untuk detail dari bagian penjadwal kunci.

Pada bagian pengacakan data, pertama-tama chiperteks $C_{(128)}$ di-XOR dengan $kw_{3(64)}||kw_{4(64)}$ dan dibagi menjadi $R_{18(64)}$and $L_{18(64)}$ dengan panjang yang sama, sehingga dapat ditulis: $C_{(128)} \oplus (kw_{3(64)}||kw_{4(64)}) = R_{18(64)}||L_{18(64)}$. Kemudian, lakukan operasi berikut dari r = 18 sampai 1, kecuali untuk r = 13 dan 7:

$$R_{r-1} = L_r \oplus F(R_r, k_r),$$
$$L_{r-1} = R_r.$$

Untuk r = 13 dan 7, dilakukan operasi sbb:

$$R'_{r-1} = L_r \oplus F(R_r, k_r),$$
$$L'_{r-1} = R_r.$$
$$R_{r-1} = FL(R'_{r-1}, kl_{2(r-1)/6}),$$
$$L_{r-1} = FL^{-1}(L'_{r-1}, kl_{2(r-1)/6-1}).$$

Terakhir, $L_{0(64)}$ dan $R_{0(64)}$ dirangkai dan di-XOR dengan $kw_{1(64)}||kw_{2(64)}$. Nilai resultannya adalah plainteks yang dapat ditulis:

$$M_{(128)} = (L_{0(64)}||R_{0(64)}) \oplus (kw_{1(64)}||kw_{2(64)}).$$

3.3.2 kunci 192 bit dan 256 bit

gambar 4 memperlihatkan prosedur dekripsi untuk kunci 192 bit atau 256 bit. Bagian pengacakan data memiliki struktur 24 tahap (round) feistel dengan 3 lapisan (layer) fungsi FL/FL^{-1} setelah tahap ke-6, ke-12 dan tahap ke-18, dan operasi XOR 128 bit sebelum tahap pertama dan setelah tahap terakhir. Bagian penjadwal kunci membangkitkan subkunci $kw_{t(64)}(t=1,2,3,4)$, $k_{u(64)}(u=1,2,...,24)$, dan $kl_{v(64)}(v=1,2,...,6)$ dari kunci rahasia K. Lihat seksi 3.4 untuk detail dari bagian penjadwal kunci.

Pada bagian pengacakan data, pertama-tama chiperteks $C_{(128)}$ di-XOR dengan $kw_{3(64)}||kw_{4(64)}$ dan dibagi menjadi $R_{24(64)}$and $L_{24(64)}$ dengan panjang yang sama, sehingga dapat ditulis: $C_{(128)} \oplus (kw_{3(64)}||kw_{4(64)}) = R_{24(64)}||L_{24(64)}$. Kemudian, lakukan operasi berikut dari r = 24 sampai 1, kecuali untuk r = 19, 13 dan 7:

$$R_{r-1} = L_r \oplus F(R_r, k_r),$$
$$L_{r-1} = R_r.$$

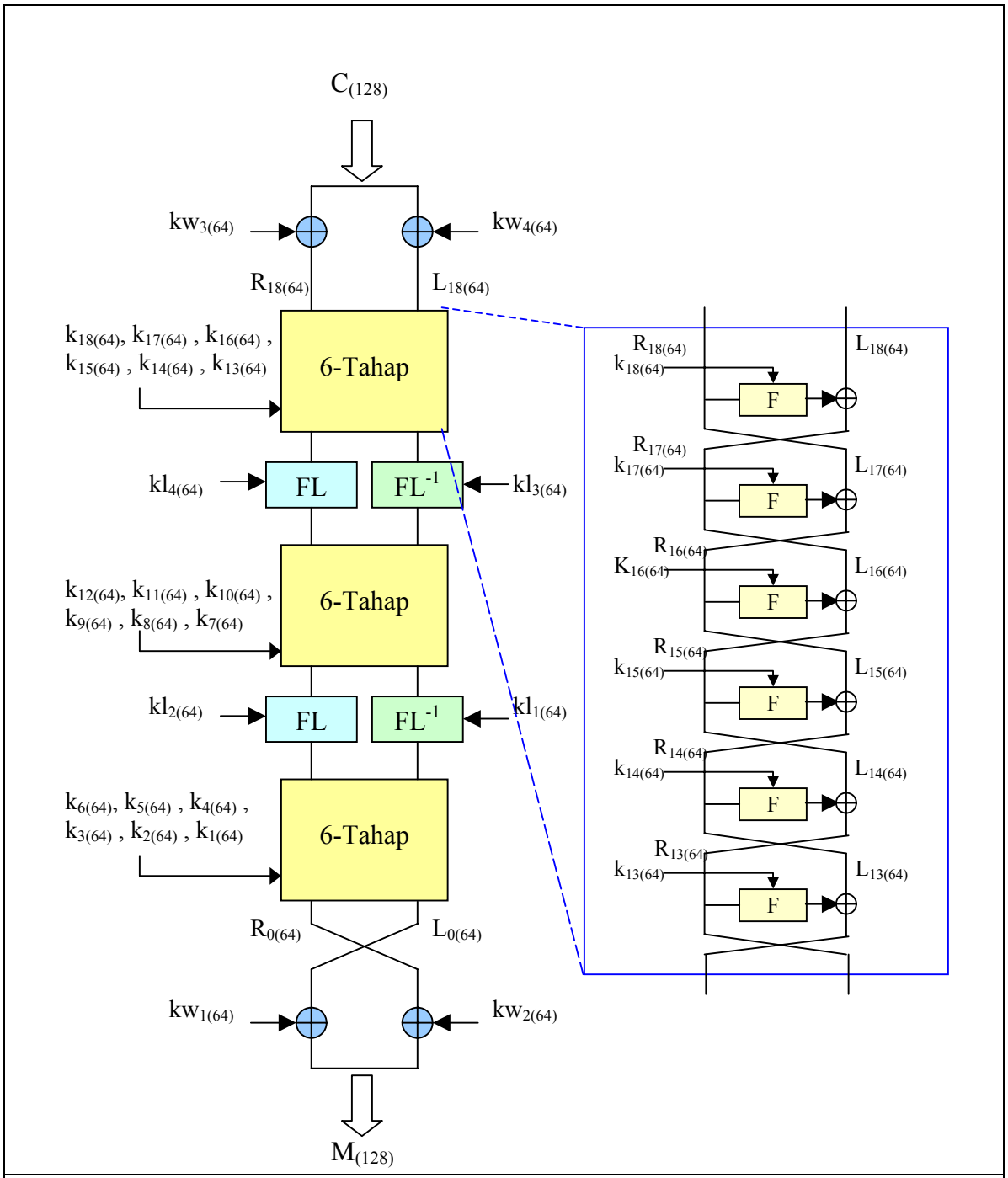
Untuk r = 19, 13 dan 7, dilakukan operasi sbb:

$$R'_{r-1} = L_r \oplus F(R_r, k_r),$$
$$L'_{r-1} = R_r.$$

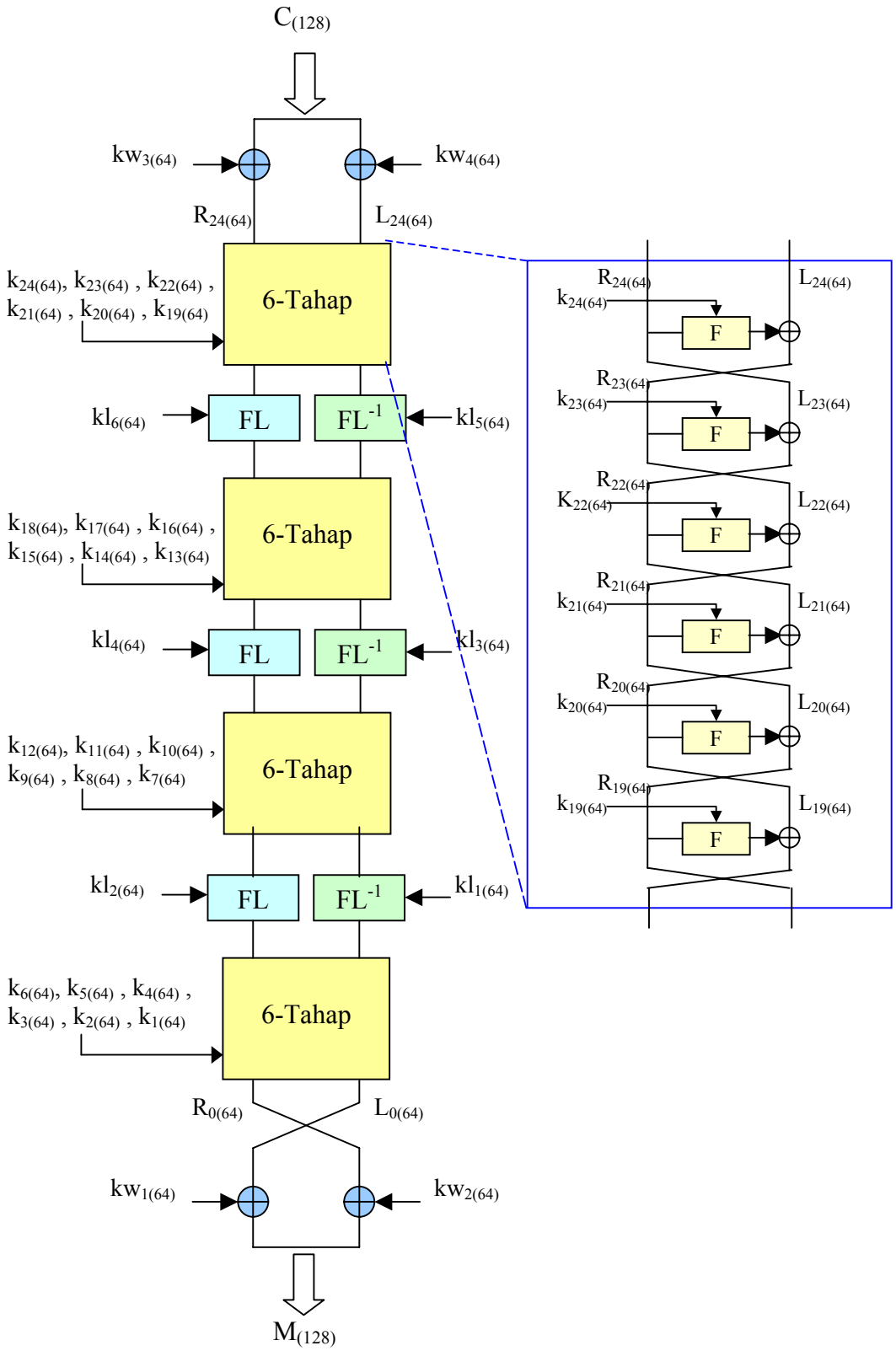
$$R_{r-1} = FL(R'_{r-1}, kl_{2(r-1)/6}),$$
$$L_{r-1} = FL^{-1}(L'_{r-1}, kl_{2(r-1)/6-1}).$$

Terakhir, $L_{0(64)}$ dan $R_{0(64)}$ dirangkai dan di-XOR dengan $kw_{1(64)}||kw_{2(64)}$. Nilai resultannya adalah plainteks yang dapat ditulis:

$$M_{(128)} = (L_{0(64)}||R_{0(64)}) \oplus (kw_{1(64)}||kw_{2(64)}).$$



Gambar 3. Prosedur Dekripsi Camellia untuk panjang kunci 128 bit.



Gambar 4. Prosedur Dekripsi Camellia untuk panjang kunci 192 dan 256 bit.

3.4 Penjadwalan kunci

Pada bagian penjadwalan kunci, diperkenalkan 2 buah variabel 128 bit $K_{L(128)}$, $K_{R(128)}$ dan empat variabel 64 bit $K_{LL(64)}$, $K_{LR(64)}$, $K_{RL(64)}$ dan $K_{RR(64)}$, yang didefinisikan sehingga dipenuhi hubungan sbb:

$K_{(128)} = K_{L(128)}$, $K_{R(128)} = 0$; untuk kunci 128 bit,
$K_{(192)} = K_{L(128)}||K_{RL(64)}$, $K_{RR(64)} = K_{RL(64)}$; untuk kunci 192 bit,
$K_{(256)} = K_{L(128)}||K_{R(128)}$; untuk kunci 256 bit.

$K_{L(128)} = K_{LL(64)}||K_{LR(64)}$, untuk semua ukuran kunci.
$K_{R(128)} = K_{RL(64)}||K_{RR(64)}$;

Dengan menggunakan variabel-variabel di atas, kita membangkitkan 2 buah variable 128 bit $K_{A(128)}$ dan $K_{B(128)}$, seperti yang dapat dilihat pada gambar 5, dimana $K_{B(128)}$ hanya dipakai ketika panjang kunci yang digunakan adalah 192 atau 256 bit. Pertama-tama, K = $K_{L(128)}$ di-XOR dengan $K_{R(128)}$ dan dienkripsi dengan 2 tahap menggunakan nilai konstanta $\Sigma_{1(64)}$ dan $\Sigma_{2(64)}$ sebagai kunci. Kemudian hasilnya di-XOR dengan $K_{L(128)}$ dan dienkripsi lagi dengan 2 tahap menggunakan nilai konstanta $\Sigma_{3(64)}$ dan $\Sigma_{4(64)}$; nilai resultannya adalah $K_{A(128)}$. Akhirnya $K_{A(128)}$ di-XOR dengan $K_{R(128)}$ dan dienkripsi lagi dengan 2 tahap menggunakan nilai konstanta $\Sigma_{5(64)}$ dan $\Sigma_{6(64)}$; Nilai resultannya adalah $K_{B(128)}$. Σi didefinisikan sebagai representasi nilai kontinyu dari tempat ke-2 heksadesimal dan tempat ke-17 heksadesimal dari akar kuadrat bilangan prima ke-i. Nilai-nilai konstanta tersebut dapat dilihat pada tabel 1.

Subkunci $kw_{t(64)}$, $k_{u(64)}$, dan $kl_{v(64)}$ dibangkitkan dari (setengah bagian kiri atau setengah bagian kanan dari) nilai hasil rotasi-penggeseran dari $K_{L(128)}$, $K_{R(128)}$, $K_{A(128)}$, dan $K_{B(128)}$. Detail subkunci diperlihatkan pada tabel 2 dan tabel 3.

Versi kunci 256 bit akan dapat mendukung (compatible) dengan versi kunci 192 bit, dengan cara mengeset nilai $K_{RR(64)}$= $K_{RL(64)}$. Versi kunci 128 bit tidak dapat didukung oleh ke-2 versi lainnya karena adanya perbedaan jumlah tahap (round) pada prosedur enkripsi dan dekripsinya.

Tabel 1. konstanta penjadwal kunci

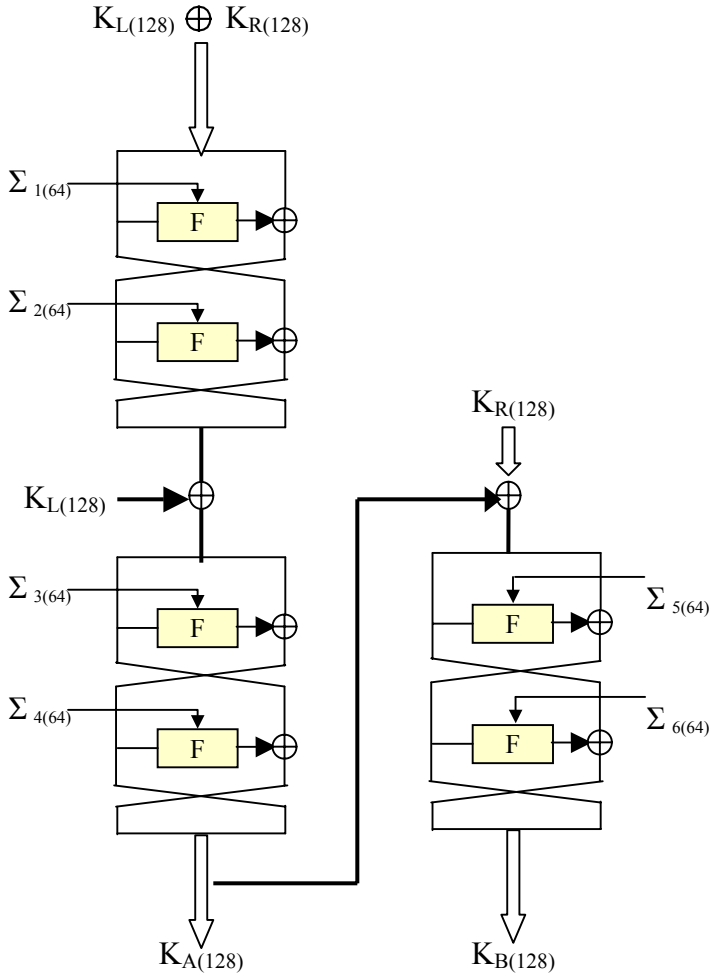
$\Sigma_{1(64)}$	0xA09E667F3BCC908B
$\Sigma_{2(64)}$	0xB67AE8584CAA73B2
$\Sigma_{3(64)}$	0xC6EF372FE94F82BE
$\Sigma_{4(64)}$	0x54FF53A5F1D36F1C
$\Sigma_{5(64)}$	0x10E527FADE682D1D
$\Sigma_{6(64)}$	0xB05688C2B3E6C1FD

Tabel 2. Subkunci untuk kunci rahasia 128 bit

	Subkunci	Nilai
Prewhitening	$Kw_{1\,(64)}$	$(KL<<<0)_{L(64)}$
	$Kw_{2(64)}$	$(KL<<<0)_{R(64)}$
F (Tahap1)	$k_{1(64)}$	$(KA<<<0)_{L(64)}$
F (Tahap2)	$k_{2\,(64)}$	$(KA<<<0)_{R(64)}$
F (Tahap3)	$k_{3(64)}$	$(KL<<<15)_{L(64)}$
F (Tahap4)	$k_{4(64)}$	$(KL<<<15)_{R(64)}$
F (Tahap5)	$k_{5(64)}$	$(KA<<<15)_{L(64)}$
F (Tahap6)	$k_{6(64)}$	$(KA<<<15)_{R(64)}$
FL	$kl_{1(64)}$	$(KA<<<30)_{L(64)}$
FL^{-1}	$kl_{2(64)}$	$(KA<<<30)_{R(64)}$
F (Tahap7)	$k_{7(64)}$	$(KL<<<45)_{L(64)}$
F (Tahap8)	$k_{8(64)}$	$(KL<<<45)_{R(64)}$
F (Tahap9)	$k_{9(64)}$	$(KA<<<45)_{L(64)}$
F (Tahap10)	$k_{10(64)}$	$(KL<<<60)_{R(64)}$
F (Tahap11)	$k_{11(64)}$	$(KA<<<60)_{L(64)}$
F (Tahap12)	$k_{12(64)}$	$(KA<<<60)_{R(64)}$
FL	$kl_{3(64)}$	$(KL<<<77)_{L(64)}$
FL^{-1}	$kl_{4(64)}$	$(KL<<<77)_{R(64)}$
F (Tahap13)	$k_{13(64)}$	$(KL<<<94)_{L(64)}$
F (Tahap14)	$k_{14\,(64)}$	$(KL<<<94)_{R(64)}$
F (Tahap15)	$k_{15(64)}$	$(KA<<<94)_{L(64)}$
F (Tahap16)	$k_{16\,(64)}$	$(KA<<<94)_{R(64)}$
F (Tahap17)	$k_{17(64)}$	$(KL<<<111)_{L(64)}$
F (Tahap18)	$k_{18(64)}$	$(KL<<<111)_{R(64)}$
Postwhitening	$Kw_{3(64)}$	$(KA<<<111)_{L(64)}$
	$Kw_{4(64)}$	$(KA<<<111)_{R(64)}$

Tabel 2. Subkunci untuk kunci rahasia 192/256 bit

	Subkunci	Nilai
Prewhitening	$kw_{1(64)}$	$(KL<<<0)_{L(64)}$
	$kw_{2(64)}$	$(KL<<<0)_{R(64)}$
F (Tahap1)	$k_{1(64)}$	$(KB<<<0)_{L(64)}$
F (Tahap2)	$k_{2(64)}$	$(KB<<<0)_{R(64)}$
F (Tahap3)	$k_{3(64)}$	$(KR<<<15)_{L(64)}$
F (Tahap4)	$k_{4(64)}$	$(KR<<<15)_{R(64)}$
F (Tahap5)	$k_{5(64)}$	$(KA<<<15)_{L(64)}$
F (Tahap6)	$k_{6(64)}$	$(KA<<<15)_{R(64)}$
FL	$kl_{1(64)}$	$(KR<<<30)_{L(64)}$
FL^{-1}	$kl_{2(64)}$	$(KR<<<30)_{R(64)}$
F (Tahap7)	$k_{7(64)}$	$(KB<<<30)_{L(64)}$
F (Tahap8)	$k_{8(64)}$	$(KB<<<30)_{R(64)}$
F (Tahap9)	$k_{9(64)}$	$(KL<<<45)_{L(64)}$
F (Tahap10)	$k_{10(64)}$	$(KL<<<45)_{R(64)}$
F (Tahap11)	$k_{11(64)}$	$(KA<<<45)_{L(64)}$
F (Tahap12)	$k_{12(64)}$	$(KA<<<45)_{R(64)}$
FL	$kl_{3(64)}$	$(KL<<<60)_{L(64)}$
FL^{-1}	$kl_{4(64)}$	$(KL<<<60)_{R(64)}$
F (Tahap13)	$k_{13(64)}$	$(KR<<<60)_{L(64)}$
F (Tahap14)	$k_{14(64)}$	$(KR<<<60)_{R(64)}$
F (Tahap15)	$k_{15(64)}$	$(KB<<<60)_{L(64)}$
F (Tahap16)	$k_{16(64)}$	$(KB<<<60)_{R(64)}$
F (Tahap17)	$k_{17(64)}$	$(KL<<<77)_{L(64)}$
F (Tahap18)	$k_{18(64)}$	$(KL<<<77)_{R(64)}$
FL	$kl_{5(64)}$	$(KA<<<77)_{L(64)}$
FL^{-1}	$kl_{6(64)}$	$(KA<<<77)_{R(64)}$
F (Tahap19)	$k_{19(64)}$	$(KR<<<94)_{L(64)}$
F (Tahap20)	$k20(64)$	$(KR<<<94)_{R(64)}$
F (Tahap21)	$k_{21(64)}$	$(KA<<<94)_{L(64)}$
F (Tahap22)	$k_{22(64)}$	$(KA<<<94)_{R(64)}$
F (Tahap23)	$k_{23(64)}$	$(KL<<<111)_{L(64)}$
F (Tahap24)	$k_{24(64)}$	$(KL<<<111)_{R(64)}$
Postwhitening	$kw_{3(64)}$	$(KB<<<111)_{L(64)}$
	$kw_{4(64)}$	$(KB<<<111)_{R(64)}$



Gambar 5. Pembangkit Kunci

3.5 Data Tes

Berikut ini adalah sampel data tes untuk Camellia (dalam heksadesimal):

kunci 128 bit

kunci	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
plainteks	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
cipherteks	67 67 31 38 54 96 69 73 08 57 06 56 48 ea be 43

kunci 192 bit

kunci	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
	00 11 22 33 44 55 66 77
plainteks	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
cipherteks	b4 99 34 01 b3 e9 96 f8 4e e5 ce e7 d7 9b 09 b9

kunci 256 bit

kunci	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
	00 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
plainteks	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
cipherteks	9a cc 23 7d ff 16 d7 6c 20 ef 7c 91 9e 3a 75 09

Data ini sering disebut juga dengan "*test vector*" yang digunakan untuk menguji apakah implementasi telah bekerja sesuai dengan spesifikasi algoritma Camellia. Data lengkap data tes dapat diperoleh di http://info.isl.ntt.co.jp/camellia/.

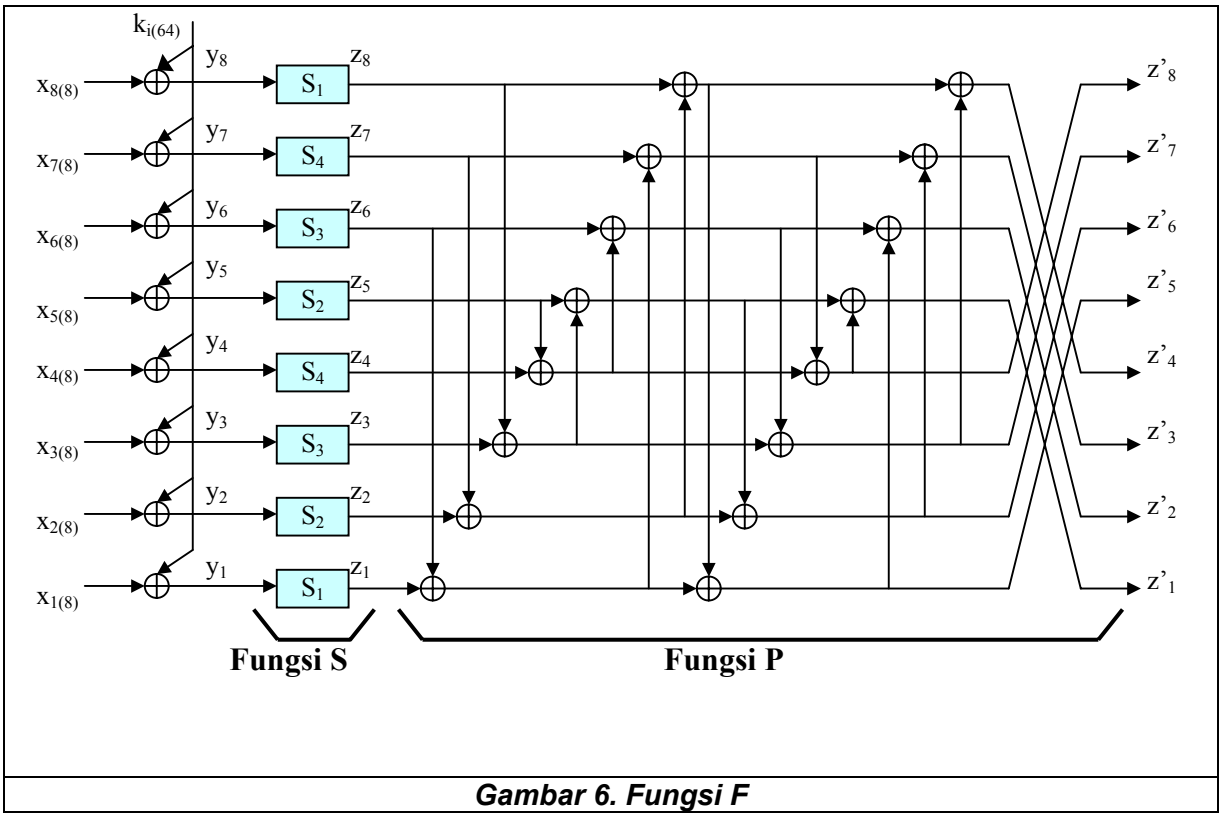
4. Komponen Penyusun Camellia

4.1 Fungsi F (F-function)

Fungsi F diperlihatkan pada gambar 6, yang didefinisikan sbb:

$$\mathbf{F : L \times L} \rightarrow \mathbf{L}$$
$$(X_{(64)}, k_{(64)}) \rightarrow Y_{(64)} = P(S(X_{(64)} \oplus k_{(64)})).$$

Lihat seksi 4.4 dan 4.6 untuk fungsi S dan fungsi P.



Gambar 6. Fungsi F

4.2 Fungsi FL

Fungsi FL diperlihatkan pada gambar 7, yang didefinisikan sbb:

$$\mathbf{F : L \times L} \rightarrow \mathbf{L}$$
$$(X_{L(32)} \| X_{R(32)}, kl_{L(32)} \| kl_{R(32)}) \rightarrow Y_{L(32)} \| Y_{R(32)},$$

dimana

$$Y_{R(32)} = ((X_{L(32)} \cap kl_{L(32)}) <<< 1 \oplus X_{R(32)},$$
$$Y_{L(32)} = (Y_{R(32)} \square\ kl_{R(32)}) \oplus X_{L(32)}.$$

4.3 Fungsi FL^{-1}

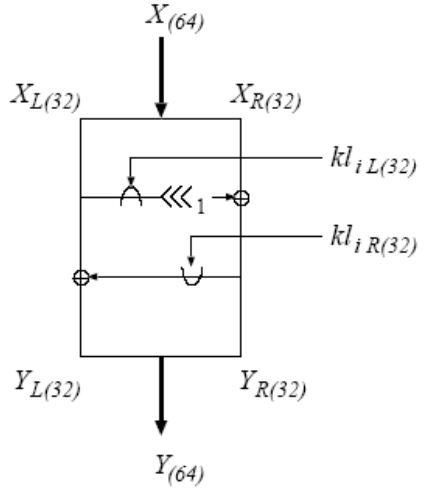
Fungsi FL^{-1} diperlihatkan pada gambar 8, yang didefinisikan sbb:

$$\mathbf{FL^{-1} : L \times L \rightarrow L}$$

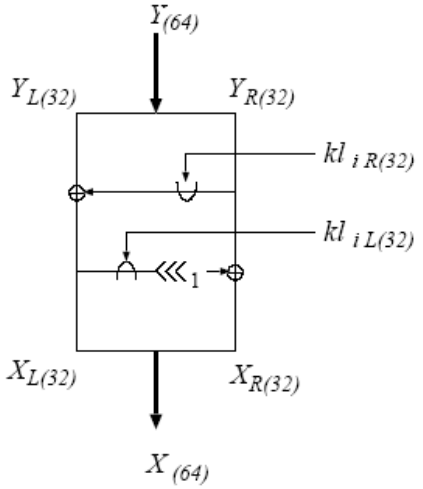
$$(Y_{L\,(32)}) \,||\, Y_{R\,(32)}\,,\, kl_{\,L\,(32)} \,||\, kl_{\,R\,(32)}) \rightarrow X_{L\,(32)} \,||\, X_{\,R\,(32)},$$

dimana:

$$X_{L\,(32)} = (Y_{\,R\,(32)} \oplus kl_{\,R\,(32)}\,) \oplus Y_{\,L\,(32)},$$
$$X_{\,R\,(32)} = ((X_{\,L\,(32)} \cap kl_{\,L\,(32)}\,) \lll 1 \oplus Y_{\,R\,(32)}$$



Gambar 7. fungsi FL



Gambar 8. fungsi FL^{-1}

4.4 Fungsi S

Fungsi S merupakan salah satu bagian dari fungsi F, yang didefinisikan sbb:

$$\mathbf{S : L \rightarrow L}$$

$$l_{1(8)} \,||\, l_{2(8)} \,||\, l_{3(8)} \,||\, l_{4(8)} \,||\, l_{5(8)} \,||\, l_{6(8)} \,||\, l_{7(8)} \,||\, l_{8(8)} \rightarrow l'_{1(8)} \,||\, l'_{2(8)} \,||\, l'_{3(8)} \,||\, l'_{4(8)} \,||\, l'_{5(8)} \,||\, l'_{6(8)} \,||\, l'_{7(8)} \,||\, l'_{8(8)}$$

$$l'_{1(8)} = s1(l_{1(8)}),$$
$$l'_{2(8)} = s2(l_{2(8)}),$$
$$l'_{3(8)} = s3(l_{3(8)}),$$
$$l'_{4(8)} = s4(l_{4(8)}),$$
$$l'_{5(8)} = s2(l_{5(8)}),$$
$$l'_{6(8)} = s3(l_{6(8)}),$$
$$l'_{7(8)} = s4(l_{7(8)}),$$
$$l'_{8(8)} = s1(l_{8(8)}),$$

dimana keempat s-box: s1, s2, s3, dan s4, dijelaskan pada seksi 4.5.

4.5 s-box

Keempat macam s-box yang dimiliki Camellia merupakan ekuivalen dari suatu fungsi invers pada $GF(2^8)$, yang ditunjukkan pada tabel 4, 5, 6, dan 7. Representasi Algebra dari keempat s-box tersebut ditunjukkan sbb:

$s_1 : B \rightarrow B$
$x_{(8)} \rightarrow h(g(f(0xc5 \oplus x(8)))) \oplus 0x6e,$
$s_2 : B \rightarrow B$
$x_{(8)} \rightarrow s_1(x_{(8)})<<<1,$
$s_3 : B \rightarrow B$
$x_{(8)} \rightarrow s_1(x_{(8)})>>>1,$
$s_4 : B \rightarrow B$
$x_{(8)} \rightarrow s_1(x_{(8)}<<<1),$

dimana fungsi f , g, dan h diberikan sbb:

f : B → B

$a_{1(1)} \| a_{2(1)} \| a_{3(1)} \| a_{4(1)} \| a_{5(1)} \| a_{6(1)} \| a_{7(1)} \| a_{8(1)} \rightarrow b_{1(1)} \| b_{2(1)} \| b_{3(1)} \| b_{4(1)} \| b_{5(1)} \| b_{6(1)} \| b_{7(1)} \| b_{8(1)}$,

dimana
$b1 = a6 \oplus a2,$
$b2 = a7 \oplus a1,$
$b3 = a8 \oplus a5 \oplus a3,$
$b4 = a8 \oplus a3,$
$b5 = a7 \oplus a4,$
$b6 = a5 \oplus a2,$
$b7 = a8 \oplus a1,$
$b8 = a6 \oplus a4.$

g : B → B

$a_{1(1)} \| a_{2(1)} \| a_{3(1)} \| a_{4(1)} \| a_{5(1)} \| a_{6(1)} \| a_{7(1)} \| a_{8(1)} \rightarrow b_{1(1)} \| b_{2(1)} \| b_{3(1)} \| b_{4(1)} \| b_{5(1)} \| b_{6(1)} \| b_{7(1)} \| b_{8(1)}$,

dimana

$$(b8+ b7\alpha + b6\alpha^2 + b5\alpha^3) + (b4+ b3\alpha + b2\alpha^2 + b1\alpha^3)\beta$$
$$= 1/((a8+ a7\alpha + a6\alpha^2 + a5\alpha^3) + (a4+ a3\alpha + a2\alpha^2 + a1\alpha^3)\beta).$$

Proses invers ini dilakukan pada $GF(2^8)$ dengan asumsi $\frac{1}{0} = 0$, dimana β adalah suatu elemen pada $GF(2^8)$ yang memenuhi $\beta^8 + \beta^6 + \beta^5 + \beta^3 + 1 = 0$, dan $\alpha = \beta^{238} = \beta^6 + \beta^5 + \beta^3 + \beta^2$ adalah suatu elemen pada $GF(2^4)$ yang memenuhi $\alpha^4 + \alpha + 1 = 0$.

h : B → B

$a_{1(1)} \| a_{2(1)} \| a_{3(1)} \| a_{4(1)} \| a_{5(1)} \| a_{6(1)} \| a_{7(1)} \| a_{8(1)} \rightarrow b_{1(1)} \| b_{2(1)} \| b_{3(1)} \| b_{4(1)} \| b_{5(1)} \| b_{6(1)} \| b_{7(1)} \| b_{8(1)}$,

dimana $b1 = a5 \oplus a6 \oplus a2$,
$b2 = a6 \oplus a2$,
$b3 = a7 \oplus a4$,
$b4 = a8 \oplus a2$,
$b5 = a7 \oplus a3$,
$b6 = a8 \oplus a1$,
$b7 = a5 \oplus a1$,
$b8 = a6 \oplus a3$.

4.6 Fungsi P

Fungsi P merupakan suatu bagian dari fungsi F, yang didefinisikan sbb:

$$\mathbf{P : L \rightarrow L}$$

$$z_{1(8)} \| z_{2(8)} \| z_{3(8)} \| z_{4(8)} \| z_{5(8)} \| z_{6(8)} \| z_{7(8)} \| z_{8(8)} \| \rightarrow z'_{1(8)} \| z'_{2(8)} \| z'_{3(8)} \| z'_{4(8)} \| z'_{5(8)} \| z'_{6(8)} \| z'_{7(8)} \| z'_{8(8)},$$

dimana

$z'_1 = z1 \oplus z3 \oplus z4 \oplus z6 \oplus z7 \oplus z8$,
$z'_2 = z1 \oplus z2 \oplus z4 \oplus z5 \oplus z7 \oplus z8$,
$z'_3 = z1 \oplus z2 \oplus z3 \oplus z5 \oplus z6 \oplus z8$,
$z'_4 = z2 \oplus z3 \oplus z4 \oplus z5 \oplus z6 \oplus z7$,
$z'_5 = z1 \oplus z2 \oplus z6 \oplus z7 \oplus z8$,
$z'_6 = z2 \oplus z3 \oplus z5 \oplus z7 \oplus z8$,
$z'_7 = z3 \oplus z4 \oplus z5 \oplus z6 \oplus z8$,
$z'_8 = z1 \oplus z4 \oplus z5 \oplus z6 \oplus z7$.

Secara ekivalen, transformasi ini dapat dituliskan dalam bentuk persamaan berikut:

$$\begin{pmatrix} z_8 \\ z_7 \\ . \\ . \\ . \\ z'_1 \end{pmatrix} \rightarrow \begin{pmatrix} z'_8 \\ z'_7 \\ . \\ . \\ . \\ z'_1 \end{pmatrix} = P \begin{pmatrix} z_8 \\ z_7 \\ . \\ . \\ . \\ z_1 \end{pmatrix}, \text{ dimana } \quad P = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Tabel 4. s-box s1

Tabel dibaca sbb: s1(0)= 112, s1(1)= 130, ..., s1(255) = 158.

112	130	44	236	179	39	192	229	228	133	87	53	234	12	174	65
35	239	107	147	69	25	165	33	237	14	79	78	29	101	146	189
134	184	175	143	124	235	31	206	62	48	220	95	94	197	11	26
166	225	57	202	213	71	93	61	217	1	90	214	81	86	108	77
139	13	154	102	251	204	176	45	116	18	43	32	240	177	132	153
223	76	203	194	52	126	118	5	109	183	169	49	209	23	4	215
20	88	58	97	222	27	17	28	50	15	156	22	83	24	242	34
254	68	207	178	195	181	122	145	36	8	232	168	96	252	105	80
170	208	160	125	161	137	98	151	84	91	30	149	224	255	100	210
16	196	0	72	163	247	117	219	138	3	230	218	9	63	221	148
135	92	131	2	205	74	144	51	115	103	246	243	157	127	191	226
82	155	216	38	200	55	198	59	129	150	111	75	19	190	99	46
233	121	167	140	159	110	188	142	41	245	249	182	47	253	180	89
120	152	6	106	231	70	113	186	212	37	171	66	136	162	141	250
114	7	185	85	248	238	172	10	54	73	42	104	60	56	241	164
64	40	211	123	187	201	67	193	21	227	173	244	119	199	128	158

Tabel 5. s-box s2

224	5	88	217	103	78	129	203	201	11	174	106	213	24	93	130
70	223	214	39	138	50	75	66	219	28	158	156	58	202	37	123
13	113	95	31	248	215	62	157	124	96	185	190	188	139	22	52
77	195	114	149	171	142	186	122	179	2	180	173	162	172	216	154
23	26	53	204	247	153	97	90	232	36	86	64	225	99	9	51
191	152	151	133	104	252	236	10	218	111	83	98	163	46	8	175
40	176	116	194	189	54	34	56	100	30	57	44	166	48	229	68
253	136	159	101	135	107	244	35	72	16	209	81	192	249	210	160
85	161	65	250	67	19	196	47	168	182	60	43	193	255	200	165
32	137	0	144	71	239	234	183	21	6	205	181	18	126	187	41
15	184	7	4	155	148	33	102	230	206	237	231	59	254	127	197
164	55	177	76	145	110	141	118	3	45	222	150	38	125	198	92
211	242	79	25	63	220	121	29	82	235	243	109	94	251	105	178
240	49	12	212	207	140	226	117	169	74	87	132	17	69	27	245
228	14	115	170	241	221	89	20	108	146	84	208	120	112	227	73
128	80	167	246	119	147	134	131	42	199	91	233	238	143	1	61

Tabel 6. s-box s3

56	65	22	118	217	147	96	242	114	194	171	154	117	6	87	160
145	247	181	201	162	140	210	144	246	7	167	39	142	178	73	222
67	92	215	199	62	245	143	103	31	24	110	175	47	226	133	13
83	240	156	101	234	163	174	158	236	128	45	107	168	43	54	166
197	134	77	51	253	102	88	150	58	9	149	16	120	216	66	204
239	38	229	97	26	63	59	130	182	219	212	152	232	139	2	235
10	44	29	176	111	141	136	14	25	135	78	11	169	12	121	17
127	34	231	89	225	218	61	200	18	4	116	84	48	126	180	40
85	104	80	190	208	196	49	203	42	173	15	202	112	255	50	105
8	98	0	36	209	251	186	237	69	129	115	109	132	159	238	74
195	46	193	1	230	37	72	153	185	179	123	249	206	191	223	113
41	205	108	19	100	155	99	157	192	75	183	165	137	95	177	23
244	188	211	70	207	55	94	71	148	250	252	91	151	254	90	172
60	76	3	53	243	35	184	93	106	146	213	33	68	81	198	125
57	131	220	170	124	119	86	5	27	164	21	52	30	28	248	82
32	20	233	189	221	228	161	224	138	241	214	122	187	227	64	79

Tabel 7. s-box s4

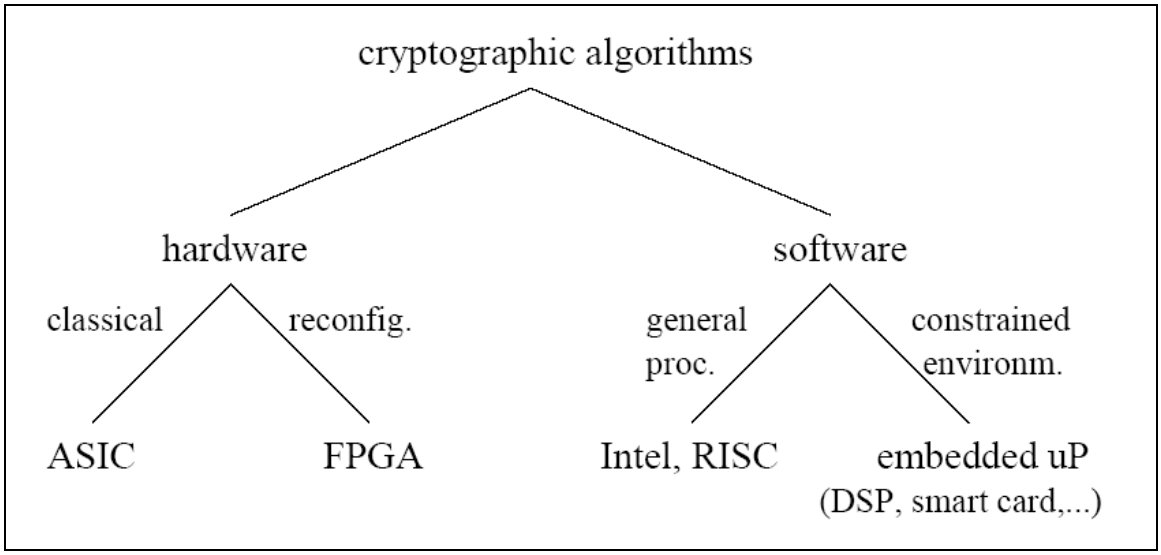
112	44	179	192	228	87	234	174	35	107	69	165	237	79	29	146
134	175	124	31	62	220	94	11	166	57	213	93	217	90	81	108
139	154	251	176	116	43	240	132	223	203	52	118	109	169	209	4
20	58	222	17	50	156	83	242	254	207	195	122	36	232	96	105
170	160	161	98	84	30	224	100	16	0	163	117	138	230	9	221
135	131	205	144	115	246	157	191	82	216	200	198	129	111	19	99
233	167	159	188	41	249	47	180	120	6	231	113	212	171	136	141
114	185	248	172	54	42	60	241	64	211	187	67	21	173	119	128
130	236	39	229	133	53	12	65	239	147	25	33	14	78	101	189
184	143	235	206	48	95	197	26	225	202	71	61	1	214	86	77
13	102	204	45	18	32	177	153	76	194	126	5	183	49	23	215
88	97	27	28	15	22	24	34	68	178	181	145	8	168	252	80
208	125	137	151	91	149	255	210	196	72	247	219	3	218	63	148
92	2	74	51	103	243	127	226	155	38	55	59	150	75	190	46
121	140	110	142	245	182	253	89	152	106	70	186	37	66	162	250
7	85	238	10	73	104	56	164	40	123	201	193	227	244	199	158

5. Implementasi

5.1 FPGA (Field Programmable Gate Array)

5.1.1 FPGA dan "*Reconfigurable Hardware*"

Dalam implementasi algoritma kriptografi, dikenal beberapa macam platform:

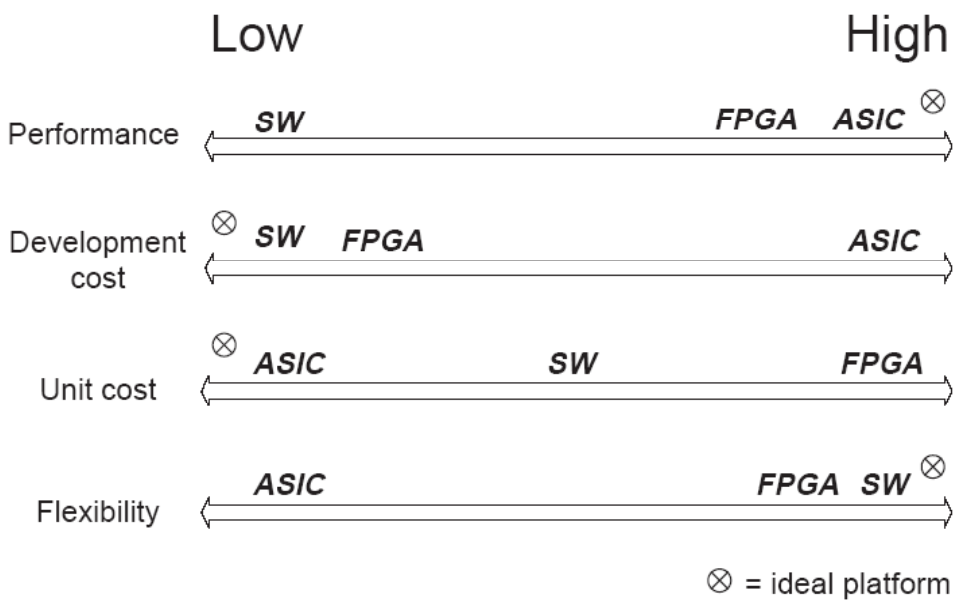


Gambar 9. Macam-macam Platform

Penentuan platform mana yang akan dipilih untuk implementasi, akan ditentukan oleh hal-hal sbb:

- Performansi Algoritma (Algorithm performance)
- Beaya / Cost: - Per-unit cost
  - Development cost
- Konsumsi Daya (Power consumption)
- Fleksibilitas
  - Parameter change
  - Key agility
  - Algorithm agility
- Keamanan Fisik (Physical security)

Platform mana yang terbaik akan ditentukan oleh kebutuhan spesifikasi dari implementasi., karakteristik dari masing masing platform akan dapat dilihat pada gambar 10.



Gambar 10. Perbandingan karakteristik berbagai platform

Hal yang menarik dari FPGA adalah bahwa pada tingkat tertentu FPGA menggabungkan kelebihan hardware dan software (lihat referensi [5]).

5.1.2 Arsitektur FPGA

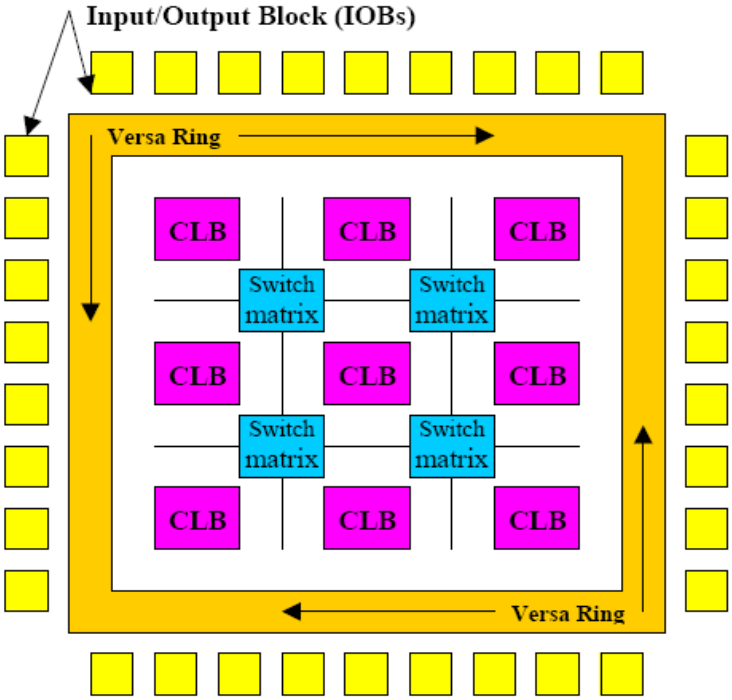
Saat ini ada 3 pemain utama dalam industri FPGA, yaitu Actel, Altera, dan Xilinx. Walaupun masing-masing jenis FPGA tersebut memiliki arsitektur yang berbeda namun secara umum FPGA akan memenuhi dan memiliki beberapa elemen kunci sbb:

- Teknologi Pemrograman (Programming technology)
- Logic sel dasar (basic logic cells)
- Input-output logic sel (I/O logic cells)
- Interkoneksi yang dapat diprogram (Programmable interconnect)
- Software untuk desain dan memrogram FPGA

Dalam tulisan ini akan dipilih implementasi dengan menggunakan Xilinx. Jenis FPGA ini(sebagai contoh xilinx 4000 series), mimiliki 4 blok penyusun utama, yaitu:

1. Configurable Logic Block (CLB)
2. Switch Matrix
3. VersaRing
4. Input/Output Block

Keempat blok tersebut digambarkan seperti pada gambar 11. Detail arsitektur FPGA tidak akan dibahas di sini.



Gambar 11. Arsitektur Xilinx 4K FPGA (chip level)

5.2 Proses Implementasi pada FPGA

Secara umum akan diperlukan tahap-tahap sbb:

1. Penentuan spesifikasi sistem
2. Perancangan arsitektur sistem
3. Perancangan blok penyusun Camellia
4. Proses coding HDL
5. simulasi, sintesis dan implementasi

5.2.1 Spesifikasi sistem

Pada tulisan ini akan diberikan sebuah sistem yang sangat sederhana sbb:

- ✓ sistem enkripsi dan dekripsi Camellia menggunakan panjang kunci 128 bit.
- ✓ Mode operasi ECB (ECB = Electronic Codebook, merupakan salah satu jenis mode operasi dari *"block chiper"*).
- ✓ Panjang input/output 128 bit

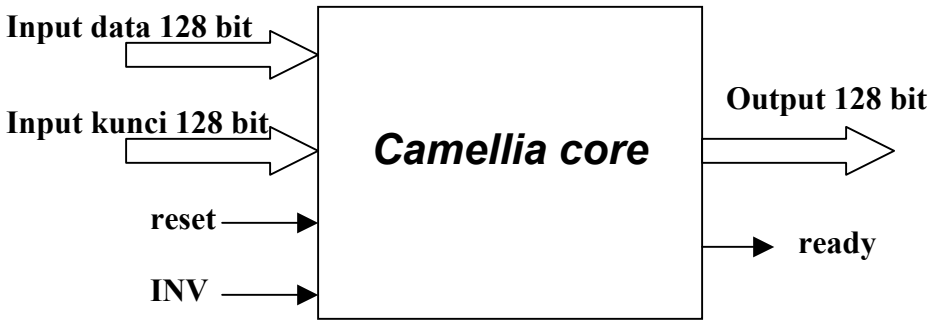
Spesifikasi sangat sederhana, namun hal ini sesuai dengan tujuannya yang hanya untuk memberikan gambaran sekilas mengenai proses implementasi camellia pada FPGA. Untuk proses implementasi yang sebenarnya, maka akan ada spesifikasi yang berkaitan dengan interface sistem tersebut dengan kondisi pemakaian yang sesungguhnya, misal: input/output memiliki lebar data 32 bit atau 64 bit.

5.2.2 Arsitektur sistem

Dengan menggunakan spesifikasi pada seksi 5.2.1 maka arsitektur sistem ini

dapat dibuat seperti pada gambar 12 dengan keterangan sbb:

- Pin reset, digunakan untuk mereset sistem
- Pin INV, digunakan untuk memilih proses enkripsi atau dekripsi
  0 = enkripsi, 1 = dekripsi
- Pin ready, digunakan untuk menadai bahwa data output telah siap



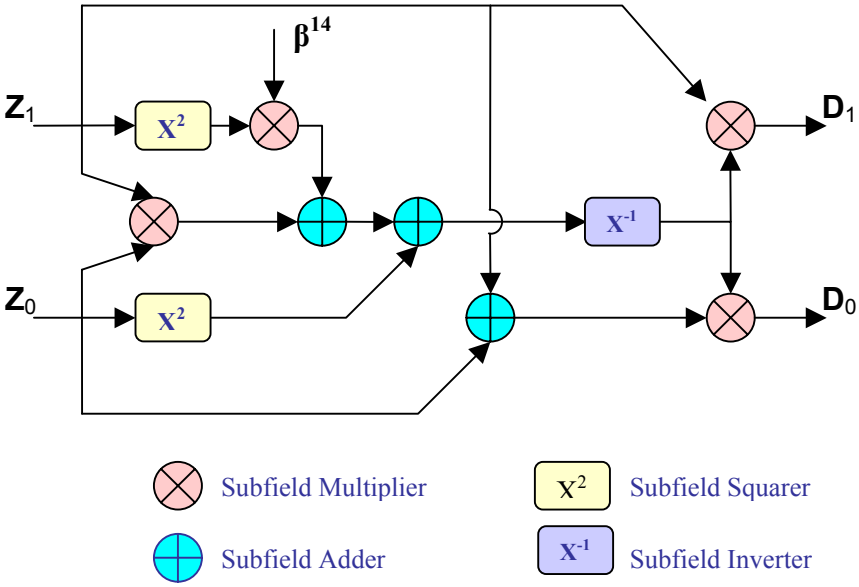
Gambar 12. Arsitektur sistem

5.2.3 Perancangan blok penyusun Camellia

Perancangan blok–blok penyusun camellia memerlukan optimasi agar implementasi yang dihasilkan memberikan hasil yang memuaskan terutama berhubungan dengan delay dan area yang dibutuhkan oleh blok tersebut.
Pengaruh blok penyusun rangkaian dapat diteliti satu persatu dari komponen-komponen penyusunnya mulai dari s-box sampai dengan rangkaian utama yang akan dipilih. Kali ini tidak akan diperinci proses perancangan blok-blok tersebut, hanya yang perlu diperhatikan bahwa komponen s-box ternyata merupakan komponen yang paling banyak digunakan dan akan menentukan ukuran besar kecilnya area pada implementasi. Sehingga penentuan blok s-box ini menjadi suatu hal yang sangat signifikan.
Dalam desain sederhana ini akan dipilih blok s-box dengan menggunakan invers multiplikatif pada composite field $GF((2^4)^2)$. Keterangan mengenai blok ini dapat dilihat secara detail pada referensi [6] (kita akan dapat dengan mudah memperoleh rangkaian s-box s1 camellia dengan mengikuti seluruh langkah pada referensi ini, kecuali penggunaan matrix S dan matrix R). Secara sederhana invers multiplikatif ini dapat dilihat pada gambar 13.
Selain itu, perlu kita perhatikan pula berapa total s-box yang akan kita gunakan dalam sistem ini. Hal ini akan mengantarkan kita kepada perhitungan berapa clock yang diperlukan untuk melakukan sebuah proses enkripsi/dekripsi secara lengkap (lihat referensi [3]).
Kita dapat memilih jumlah clock, misal:

1. 1 clock – dilakukan 1 tahap enkripsi/dekripsi
2. 1 clock – dilakukan 3 tahap enkripsi/dekripsi
3. 1 clock – dilakukan 6 tahap enkripsi/dekripsi

Pemilihan hal di atas akan berpengaruh kepada kecepatan dan area yang dihasilkan oleh implementasi. Pada seksi selanjutnya akan diperlihatkan perbandingan antara opsi 2 dan 3



Gambar 13. Invers Multiplikatif pada composite field GF((2⁴)²)

5.2.4 Coding HDL

Proses selanjutnya setelah menetukan spesifikasi dan arsitektur sistem adalah memodelkan hasil perancangan dalam bentuk HDL (*Hardware Description Language*).ada 2 macam HDL yaitu verilog dan VHDL.

Baik buruknya penyusunan HDL akan sangat berpengaruh kepada kualitas hasil implementasi. HDL akan sangat baik bila disusun dalam tingkat RTL (*register transfer level*). Tutorial tentang VHDL maupun verilog dapat dengan mudah ditemukan di internet. Dalam tulisan ini penulis cenderung menggunakan VHDL daripada verilog.

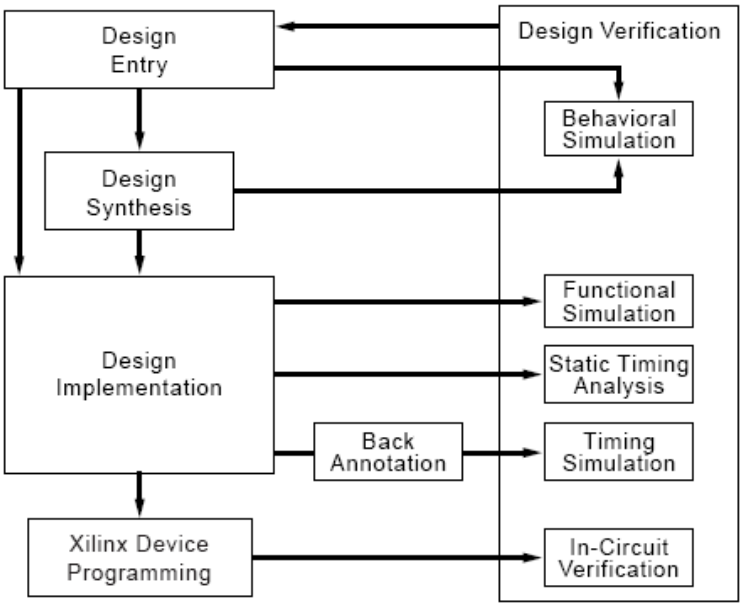
5.2.5 Simulasi, sintesis dan implementasi

Dalam tahap ini dugunakan beberapa software bantu (tools) yaitu sbb:

- ISE 6.2i., sebagai software bantu utama
- ModelSim 5.7g XE starter edition, digunakan untuk simulasi (terintegrasi pada lingkungan pengembangan ISE 6.2i)
- Xilinx Synthesis Technology (terintegrasi pada ISE 6.2i), untuk melakukan proses "syntesis".

Tahap implementasi pada xilinx dapat digambarkan sebagai diagram alir seperti terlihat pada gambar 14. Proses design entry dapat dilakukan baik dengan pendekatan skematik maupun dengan menggunakan VHDL. Desain yang pertama-tama dibuat (skematik / VHDL) kemudian akan diuji kebenaran logikanya dengan simulasi behavioral (pada tahap ini parameter waktu seperti

delay belum diperhitungkan). Proses sintesis dilakukan untuk melihat besarnya ukuran desain (dari resource yang dihabiskan oleh desain) serta informasi delay yang dihasilkan oleh desain. Selanjutnya informasi tentang resource yang ada digunakan untuk mengevaluasi apakah desain ini layak untuk diteruskan ataukah harus kita desain kembali. Sedangkan informasi tentang delay akan kita gunakan untuk menentukan clock yang akan kita gunakan dalam sistem yang kita desain.



Gambar 14. Diagram alir proses implementasi pada xilinx

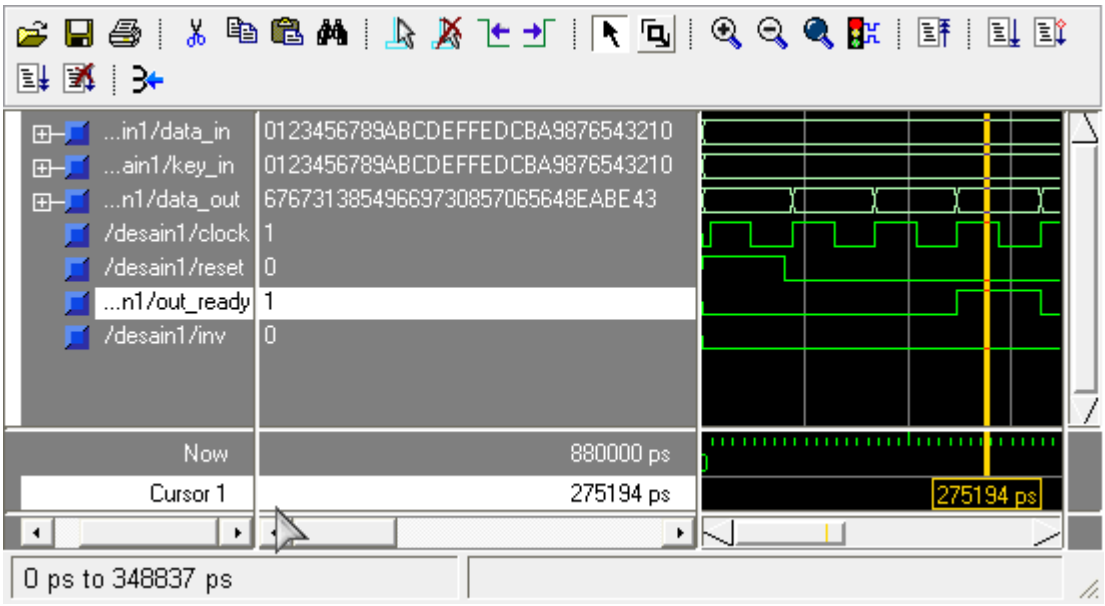
Proses Implementasi selanjutnya akan mengevaluasi desain dengan parameter yang lebih lengkap seperti delay misalnya (tidak seperti pada simulasi behavioral). Setelah desain melalui tahap-tahap tersebut maka selanjutnya desain akan didownload ke device dan akan diverifikasi apakah telah sesuai dengan spesifikasi. Bila telah sesuai dengan spesifikasi yang diinginkan, maka desain kita telah selesai, dan bila belum, maka kita kita harus mengulangnya dengan memperbaiki kekurangan yang ada.
Dalam tuisan ini akan ditunjukkan contoh proses sampai dengan sintesis.
Contoh: Sesuai dengan apa yang telah ditulis pada seksi 5.2.3 maka desain yang telah kita buat pada seksi sebelumnya akan disimulasikan secara behavioral dan disintesis. Akan kita coba 2 macam perhitungan clock:

- Desain 1: 1 clock – dilakukan 6 tahap enkripsi/dekripsi
  Sehingga akan diperlukan 1 clock untuk proses setup, dan 3 clock untuk enkripsi/dekripsi (ingat bahwa desain dilakukan pada panjang kunci 128 bit dan diperlukan total 18 tahap untuk melakukan enkripsi/dekripsi tanpa menghitung proses setup kunci)
- Desain 2: 1 clock – dilakukan 3 tahap enkripsi/dekripsi
  Sehingga akan diperlukan 2 clock untuk proses setup, dan 6 clock untuk enkripsi/dekripsi

Simulasi Behavioral

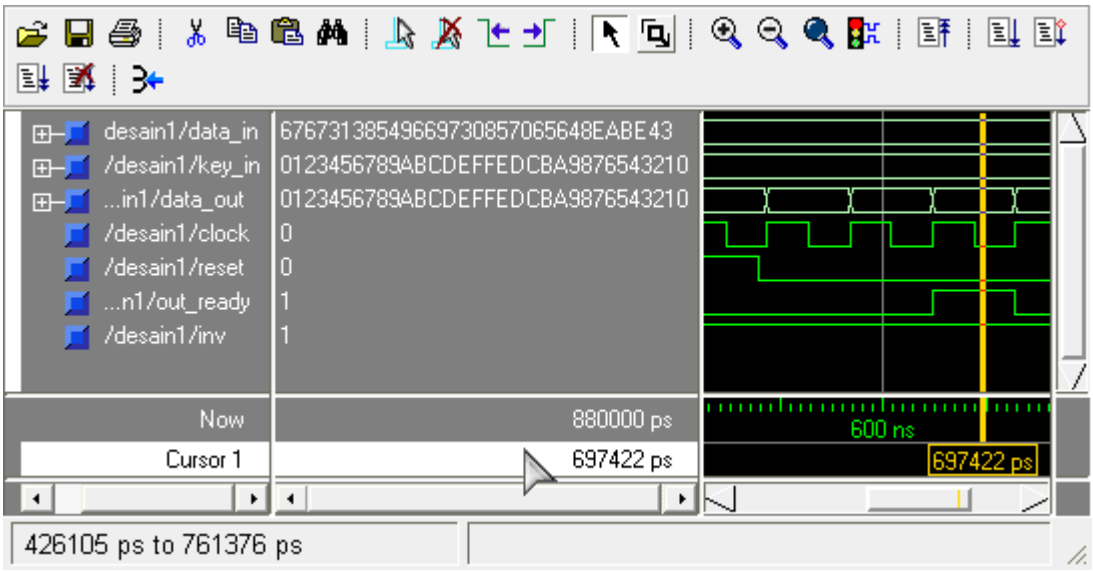
Digunakan vektor uji (kunci 128 bit) sbb:

kunci	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
plainteks	01 23 45 67 89 ab cd ef fe dc ba 98 76 54 32 10
cipherteks	67 67 31 38 54 96 69 73 08 57 06 56 48 ea be 43

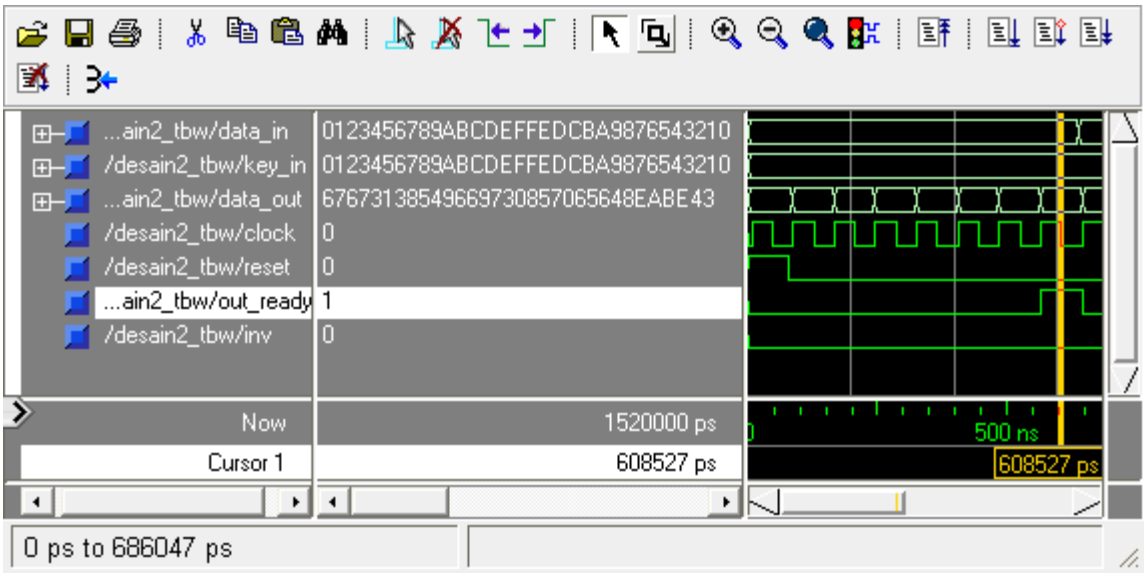


Gambar 15. Simulasi enkripsi desain 1

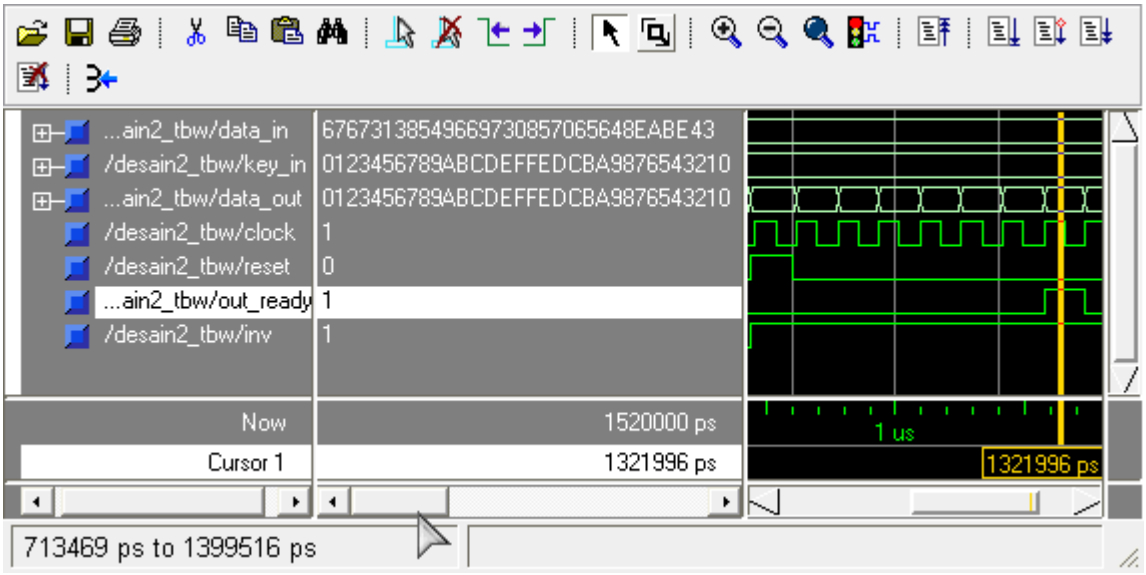
(1 clock untuk set up, 3 clock untuk proses enkripsi)



Gambar 16.Simulasi dekripsi desain1



Gambar 17.Simulasi enkripsi desain 2
(2 clock untuk set up, 6 clock untuk proses enkripsi)



Gambar 18.Simulasi dekripsi desain 2

Dari gambar-gambar di atas dapat terlihat bahwa simulasi behavioral sistem telah memenuhi ciri algoritma Camellia (sesuai dengan vektor uji yang dipakai).

Sintesis

Hasil sintesis adalah sbb:

Desain 1:

```
. . . . . .
Total equivalent gate count for design:  55,583
Additional JTAG gate count for IOBs:  18,624
Peak Memory Usage:  144 MB
. . . . . .

Device utilization summary:
---------------------------
Selected Device : 2v2000bf957-6

 Number of Slices:                  4509  out of  10752    41%
 Number of Slice Flip Flops:         391  out of  21504     1%
 Number of 4 input LUTs:            8677  out of  21504    40%
 Number of bonded IOBs:              387  out of    624    62%
 Number of GCLKs:                      1  out of     16     6%
. . . . . .
Timing Summary:
---------------
Speed Grade: -6

   Minimum period: 52.029ns (Maximum Frequency: 19.220MHz)
   Minimum input arrival time before clock: 53.639ns
   Maximum output required time after clock: 56.591ns
   Maximum combinational path delay: 58.201ns
 . . . . . .
```

Desain 2:

```
. . . . . .
Total equivalent gate count for design:  43,761
Additional JTAG gate count for IOBs:  18,624
Peak Memory Usage:  134 MB
. . . . . .

Device utilization summary:
---------------------------
Selected Device : 2v2000bf957-6

 Number of Slices:                  3445  out of  10752    32%
 Number of Slice Flip Flops:         471  out of  21504     2%
 Number of 4 input LUTs:            6575  out of  21504    30%
 Number of bonded IOBs:              387  out of    624    62%
 Number of GCLKs:                      1  out of     16     6%
. . . . . .
Timing Summary:
---------------
Speed Grade: -6

   Minimum period: 28.792ns (Maximum Frequency: 34.732MHz)
   Minimum input arrival time before clock: 29.789ns
   Maximum output required time after clock: 33.354ns
```

```
    Maximum combinational path delay: 34.351ns
. . . . . ..
```

Bila dirangkum dalam sebuah tabel adalah sbb:

Tabel 8. Perbandingan hasil sintesis

No	Nama	Desain1	Desain2
1	Devais	xc2v2000bf957-6	xc2v2000bf957-6
2	Slices	4509	3445
3	Slices Flip-flops	391	471
4	4 input LUTs	8677	6575
5	bonded IOBs	387	387
6	GCLKs	1	1
7	equivalent gate	55,583	43,761
8	JTAG gate	18,624	18,624
9	Minimum period	52.029ns	28.792ns
10	Maximum frequency	19.220MHz	34.732MHz
11	Maximum comb. delay	58.201ns	34.351ns
12	Maximum Throughput*	820 Mbit/s	741 Mbit/s
13	keterangan	1 clock setup,3 clock enkripsi/dekripsi	2 clock setup, 6 clock enkripsi/dekripsi

*nb: throughput = (lebar data x frekuenci kerja) / jumlah cycle enkripsi 1 blok data

Analisis:

Dari teks hasil sintesis dapat kitalihat bahwa ukuran device target masih jauh lebih besar daripada ukuran resource yang dibituhkan oleh desain sehingga dapat diambil kesimpulan bahwa desain ini akan mampu untuk diimplementasikan pada device target **Xilinx `xc2v2000bf957-6.`**

Terlihat dari tabel perbandingan bahwa desain pertama membutuhkan resource yang lebih besar daripada desain 2, sedangkan throughput maksimum desain 1 lebih besar daripada desain 2.Hal ini dapat dijelaskan sbb:

- Ukuran resource
  Kembali lagi ke masalah pemakaian s-box, bila dihitung maka jumlah s-box yang dipakai oleh desain 1 lebih banyak daripada yang dipakai oleh desain 2. Satu tahap enkripsi memakai 8 buah s-box (lihat referensi [3]) sehingga:
  - desain 1 (1 clock- 6 tahap) akan memakai 6x8 s-box = 68 s-box
  - desain 2 (1 clock- 3 tahap) akan memakai 3x8 s-box = 24 s-box
- Throughput
  Hal ini disebabkan oleh jumlah clock yang berbeda. Dalam kasus ini, peningkatan "maximum frequency" pada desain 2 tidak dapat mengimbangi penurunan "minimum period" yang terjadi pada desain 2.

6. Kesimpulan

1. Camellia merupakan sebuah Algoritma Kriptografi yang tergolong pada tipe algoritma kunci simetris.
2. Camellia mendukung ukuran blok data 128 bit dengan panjang kunci 128-bit, 192-bit, atau 256-bit.
3. Implementasi Camellia pada platform FPGA akan menjumpai beberapa "*constraint*" diataranya adalah ukuran "*resource*" dan "*speed*" (kecepatan proses) dari hasil implementasi.
4. Penentuan arsitektur serta blok penyusun sistem pada proses perancangan akan sangat berpengaruh kepada performansi implementasi Camellia pada FPGA.

7. Referensi

[1] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita. *Camellia a 128-bit block cipher suitable for multiple platforms.* NTT and Mitsubishi Electric Corporation, 2000.Tersedia di http://info.isl.ntt.co.jp/camellia/.

[2] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita. Specification of Camellia - *a 128-bit block cipher*. NTT and Mitsubishi Electric Corporation, 2000. Tersedia di http://info.isl.ntt.co.jp/camellia/

[3] Rogawski, Marcin. A*nalysis of Implementation of Hierocrypt–3 Algorithm (and its comparison to camellia algorithm) using altera devices***. M**ilitary University of Technology Institute of Mathematics and Cryptology Faculty of Cybernetics, 2003 .

[4] Yiqun Lisa Yin. *A Note on the Block Cipher Camellia.* NTT Multimedia Communication Laboratories, 2000.

[5] Paar, Christof. *Reconfigurable Hardware in Modern Cryptography.* Cryptography and Information Security Group Electrical & Computer Engineering Dept. and Computer Science Dept. Worcester Polytechnic Institute Worcester, MA, USA http://www.ece.wpi.edu/Research/crypt

[6] O'Driscoll, Cillian. *A Note on Composite Galois Fields and their application to Rijndael.* 2001.