1. ファイル一覧

File name	Type	Programming	Explanation
HyRAL.c	Source	ANSI C	HyRAL暗号／復号機能のソースコード
HyRAL.h	Header	ANSI C	HyRAL暗号／復号機能のヘッダコード

2. コンパイルオプション一覧

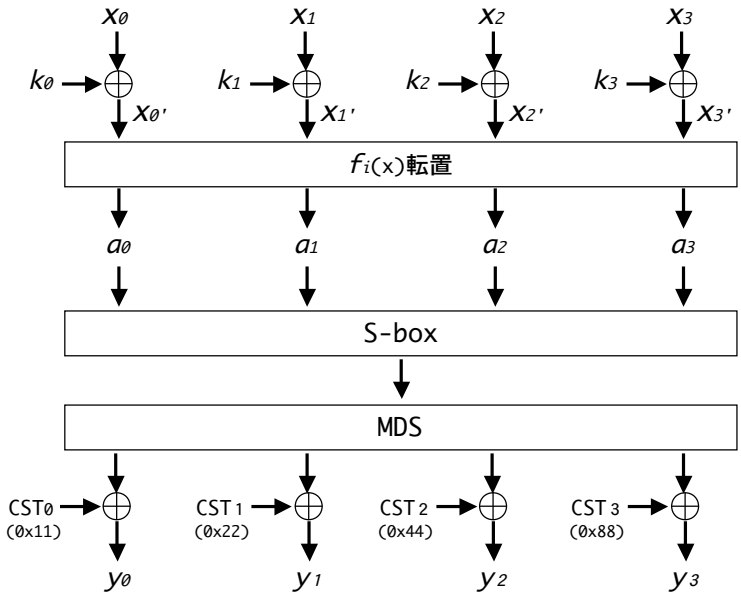
Option name	Explanation
__BIG_ENDIAN__	このオプションを指定する事により、ビックエンディアンに対応したHyRAL暗号／復号を実現する。 ※CPUタイプを自動判定しオプション「__BIG_ENDIAN__」もしくは「__LITTLE_ENDIAN__」が設定されます。
__LITTLE_ENDIAN__	このオプションを指定する事により、リトルエンディアンに対応したHyRAL暗号／復号を実現する。 ※CPUタイプを自動判定しオプション「__BIG_ENDIAN__」もしくは「__LITTLE_ENDIAN__」が設定されます。
__STANDARD_COMP__	このオプションが指定されている場合、「f関数、F関数、G関数」は標準関数としてビルドされます。 このオプションが指定されていない場合、「f関数、F関数、G関数」はマクロ関数（速度重視）としてビルドされます。
__TEST_VECTOR__	このオプションはテストベクトル用です。 オプションが指定された場合、「f関数、F関数、G関数」の結果が出力されます。 また「__TEST_VECTOR__」が設定されている場合、自動で「__STANDARD_COMP__」が設定されます。

3. 関数一覧

No	Function name	Explanation
1	f1 f2 f3 f4 f5 f6 f7 f8	暗号／復号の共有機能。 f関数テーブルからデータを引き出す機能。
2	KeyGenerationSingle KeyGenerationDouble	入力Keyを元にKeyMaterialを生成する keyGenerationSingle:128bitKeyモード、KeyGenerationDouble:192/256bitKeyモード
3	KeySchedulingSingle KeySchedulingDouble	KeyMaterialを利用してSubKeyを生成する KeySchedulingSingle:128bitモード、KeySchedulingDouble:192/256bitKeyモード
4	F1 F2	暗号化処理の一部機能。
5	G1 G2	暗号化処理の一部機能。
6	HyralSingle HyralDouble	HyRAL暗号化のメインモジュール HyRALSingle:128bitKey暗号機能、HyRALDouble:192/256bitKey暗号機能
7	rF1 rF2	復号化処理の一部機能。
8	rG1 rG2	復号化処理の一部機能。
9	rHyralSingle rHyralDouble	HyRAL復号化のメインモジュール rHyRALSingle:128bitKey復号機能、rHyRALDouble:192/256bitKey復号機能
-	init ClearKM ClearIK ClearRK	KeyMaterial（KM）、SubKey（IK、RK）等の変数をクリアする処理
-	getKM getIK getRK	KeyMaterial（KM）、SubKey（IK、RK）等の値を取得する ※テスト用関数
-	TestVectorPrint32bit TestVectorPrint128bit	データを出力する機能 ※テスト用関数

4. 関数説明

4.1 f関数



4.1.1 f1関数

f1...(x0',x1',x2',x3')

構文

unsigned long f1(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

※コンパイル・オプションによっては、マクロ関数になる。

4.1.2 f2関数

f2...(x1',x2',x3',x0')

構文

unsigned long f2(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

※コンパイル・オプションによっては、マクロ関数になる。

4.1.3 f3関数

f3...(x2',x3',x0',x1')

構文

unsigned long f3(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

※コンパイル・オプションによっては、マクロ関数になる。

4.1.4 f4関数

f4...(x3',x0',x1',x2')

構文

unsigned long f4(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

※コンパイル・オプションによっては、マクロ関数になる。

4.1.5 f5関数

f5...(x3',x2',x1',x0')

構文

unsigned long f5(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

※コンパイル・オプションによっては、マクロ関数になる。

4.1.6 f6関数

f6...(x2',x1',x0',x3')

構文

unsigned long f6(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

※コンパイル・オプションによっては、マクロ関数になる。

4.1.7 f7関数

f7...(x1',x0',x3',x2')

構文

unsigned long f7(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

※コンパイル・オプションによっては、マクロ関数になる。

4.1.8 f8関数

f8...(x0',x3',x2',x1')

構文

unsigned long f8(const unsigned long Im)

引数

Im [in] f関数テーブルからデータを索引する為の元データ [32bit]

戻り値

引数を元にf関数テールを読みデータを転置した値を戻す

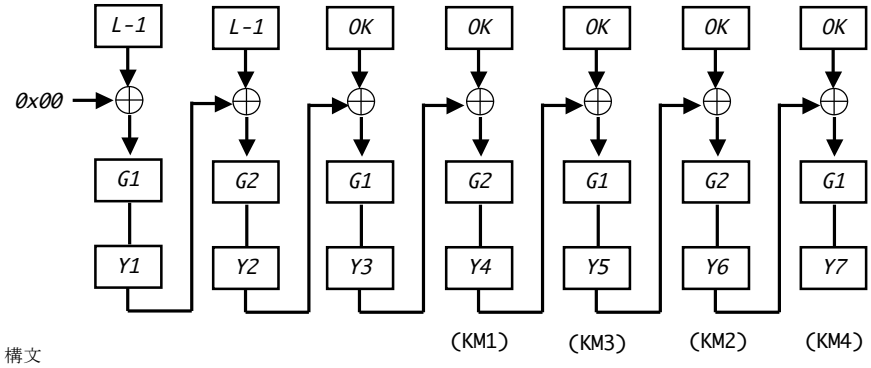
※コンパイル・オプションによっては、マクロ関数になる。

4.2 KeyGeneration関数

オリジナルKeyを元にKey Materialを生成する

4.2.1 KeyGenerationSingle関数

Single Key Mode (Original Key[OK] 128bit)



構文

void KeyGenerationSingle(unsigned long* OK)

引数

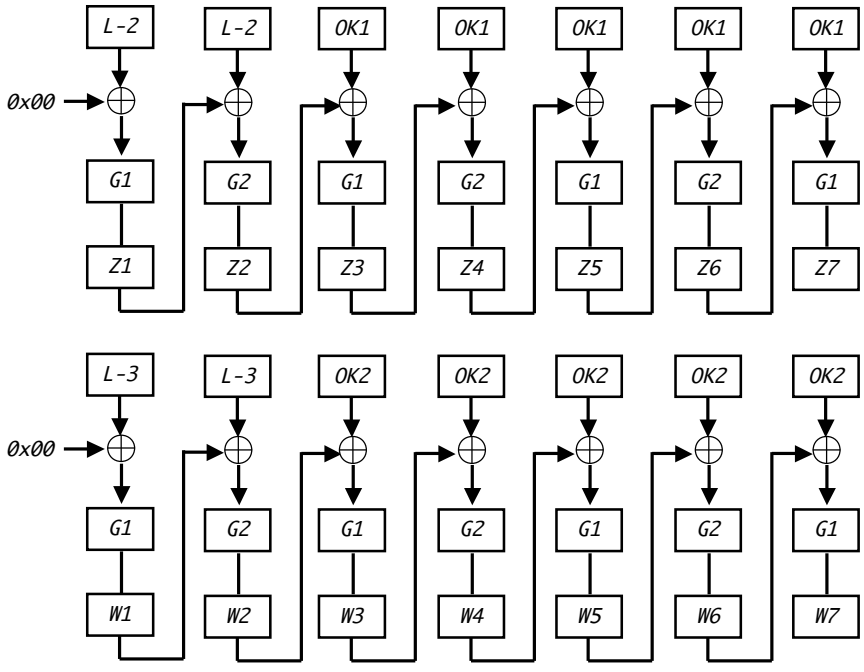
OK	[in]	暗号／復号Key[128bit]

戻り値

なし

4.2.2 KeyGenerationDouble関数

Double Key Mode (Original Key1[OK1] 128bit、Key2[OK2] 128bit)



Double Key Modeの場合
KM1 = Z4 xor W4
KM2 = Z6 xor W6
KM3 = Z5 xor W5
KM4 = Z7 xor W7

構文

void KeyGenerationSingle(unsigned long* OK)

引数

OK1	[in]	オリジナルKey（暗号／復号Key）[128bit]
OK2	[in]	オリジナルKey（暗号／復号Key）[128bit]

戻り値

なし

4.3 KeyScheduling関数

「4.2 KeyGeneration」で生成したKeyMaterial(KMi)を使用してSubKeyを作成する

4.3.1 KeySchedulingSingle関数

Single Key Mode

SubKey	0ブロック	1ブロック	2ブロック	3ブロック
RK4	RK4[0] = KM1[0] xor KM4[3]	RK4[1] = KM1[1] xor KM4[2]	RK4[2] = KM1[2] xor KM4[1]	RK4[3] = KM1[3] xor KM4[0]
IK1	IK1[0] = KM1[1] xor KM4[0]	IK1[1] = KM1[2] xor KM4[3]	IK1[2] = KM1[3] xor KM4[2]	IK1[3] = KM1[0] xor KM4[1]
IK4	IK4[0] = KM1[2] xor KM4[0]	IK4[1] = KM1[3] xor KM4[1]	IK4[2] = KM1[0] xor KM4[2]	IK4[3] = KM1[1] xor KM4[3]
RK1	RK1[0] = KM3[3] xor KM4[0]	RK1[1] = KM3[2] xor KM4[1]	RK1[2] = KM3[1] xor KM4[2]	RK1[3] = KM3[0] xor KM4[3]
RK6	RK6[0] = KM3[0] xor KM4[1]	RK6[1] = KM3[3] xor KM4[2]	RK6[2] = KM3[2] xor KM4[3]	RK6[3] = KM3[1] xor KM4[0]
RK3	RK3[0] = KM3[0] xor KM4[2]	RK3[1] = KM3[1] xor KM4[3]	RK3[2] = KM3[2] xor KM4[0]	RK3[3] = KM3[3] xor KM4[1]
IK2	IK2[0] = KM3[3] xor KM4[3]	IK2[1] = KM3[0] xor KM4[0]	IK2[2] = KM3[1] xor KM4[1]	IK2[3] = KM3[2] xor KM4[2]
RK7	RK7[0] = KM1[3] xor KM2[0]	RK7[1] = KM1[2] xor KM2[1]	RK7[2] = KM1[1] xor KM2[2]	RK7[3] = KM1[0] xor KM2[3]
RK2	RK2[0] = KM1[0] xor KM2[1]	RK2[1] = KM1[3] xor KM2[2]	RK2[2] = KM1[2] xor KM2[3]	RK2[3] = KM1[1] xor KM2[0]
RK5	RK5[0] = KM1[0] xor KM2[2]	RK5[1] = KM1[1] xor KM2[3]	RK5[2] = KM1[2] xor KM2[0]	RK5[3] = KM1[3] xor KM2[1]
IK3	IK3[0] = KM1[3] xor KM2[3]	IK3[1] = KM1[0] xor KM2[0]	IK3[2] = KM1[1] xor KM2[1]	IK3[3] = KM1[2] xor KM2[2]

構文

void KeySchedulingSingle()

引数

なし

戻り値

なし

4.3.2 KeySchedulingDouble関数

Double Key Mode

SubKey	0ブロック	1ブロック	2ブロック	3ブロック
RK5	RK5[0] = KM1[0] xor KM4[3]	RK5[1] = KM1[1] xor KM4[2]	RK5[2] = KM1[2] xor KM4[1]	RK5[3] = KM1[3] xor KM4[0]
IK1	IK1[0] = KM1[1] xor KM4[0]	IK1[1] = KM1[2] xor KM4[3]	IK1[2] = KM1[3] xor KM4[2]	IK1[3] = KM1[0] xor KM4[1]
IK6	IK6[0] = KM1[2] xor KM4[0]	IK6[1] = KM1[3] xor KM4[1]	IK6[2] = KM1[0] xor KM4[2]	IK6[3] = KM1[1] xor KM4[3]
RK1	RK1[0] = KM3[3] xor KM4[0]	RK1[1] = KM3[2] xor KM4[1]	RK1[2] = KM3[1] xor KM4[2]	RK1[3] = KM3[0] xor KM4[3]
RK8	RK8[0] = KM3[0] xor KM4[1]	RK8[1] = KM3[3] xor KM4[2]	RK8[2] = KM3[2] xor KM4[3]	RK8[3] = KM3[1] xor KM4[0]
RK3	RK3[0] = KM3[0] xor KM4[2]	RK3[1] = KM3[1] xor KM4[3]	RK3[2] = KM3[2] xor KM4[0]	RK3[3] = KM3[3] xor KM4[1]
RK4	RK4[0] = KM3[3] xor KM4[3]	RK4[1] = KM3[0] xor KM4[0]	RK4[2] = KM3[1] xor KM4[1]	RK4[3] = KM3[2] xor KM4[2]
RK9	RK9[0] = KM1[3] xor KM2[0]	RK9[1] = KM1[2] xor KM2[1]	RK9[2] = KM1[1] xor KM2[2]	RK9[3] = KM1[0] xor KM2[3]
RK2	RK2[0] = KM1[0] xor KM2[1]	RK2[1] = KM1[3] xor KM2[2]	RK2[2] = KM1[2] xor KM2[3]	RK2[3] = KM1[1] xor KM2[0]
RK7	RK7[0] = KM1[0] xor KM2[2]	RK7[1] = KM1[1] xor KM2[3]	RK7[2] = KM1[2] xor KM2[0]	RK7[3] = KM1[3] xor KM2[1]
RK6	RK6[0] = KM1[3] xor KM2[3]	RK6[1] = KM1[0] xor KM2[0]	RK6[2] = KM1[1] xor KM2[1]	RK6[3] = KM1[2] xor KM2[2]
IK2	IK2[0] = KM2[3] xor KM3[0]	IK2[1] = KM2[2] xor KM3[1]	IK2[2] = KM2[1] xor KM3[2]	IK2[3] = KM2[0] xor KM3[3]
IK3	IK3[0] = KM2[0] xor KM3[1]	IK3[1] = KM2[3] xor KM3[2]	IK3[2] = KM2[2] xor KM3[3]	IK3[3] = KM2[1] xor KM3[0]
IK4	IK4[0] = KM2[0] xor KM3[2]	IK4[1] = KM2[1] xor KM3[3]	IK4[2] = KM2[2] xor KM3[0]	IK4[3] = KM2[3] xor KM3[1]
IK5	IK5[0] = KM2[3] xor KM3[3]	IK5[1] = KM2[0] xor KM3[0]	IK5[2] = KM2[1] xor KM3[1]	IK5[3] = KM2[2] xor KM3[2]

構文

void KeySchedulingDouble()

引数

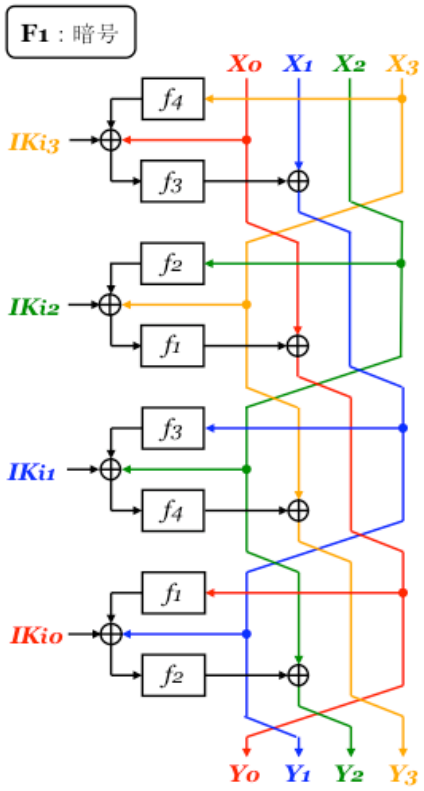
なし

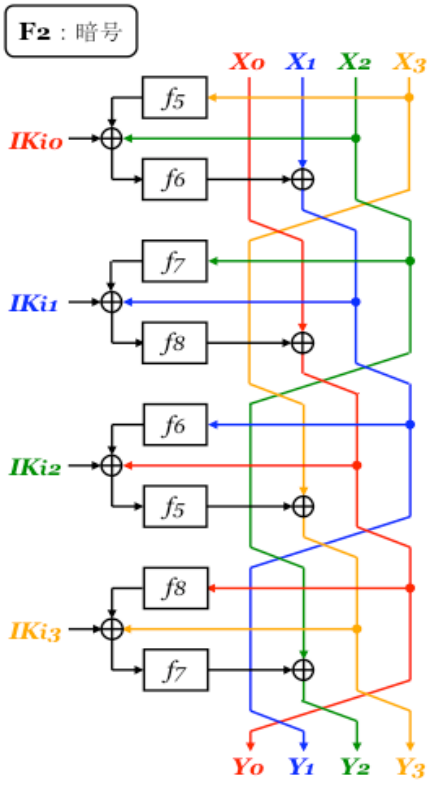
戻り値

なし

4.4 F関数

暗号化処理の一部機能





4.4.1　F1関数

構文

```
void F1(unsigned long* XtoY, unsigned long* IKi)
```

引数

XtoY	[in/out]	データをf関数とSubKeyを利用して暗号化して戻す［128bit］
IKi	[in]	SubKey

戻り値

なし

※コンパイル・オプションによっては、マクロ関数になる。

4.4.2　F2関数

構文

```
void F2(unsigned long* XtoY, unsigned long* IKi)
```

引数

XtoY	[in/out]	データをf関数とSubKeyを利用して暗号化して戻す［128bit］
IKi	[in]	SubKey

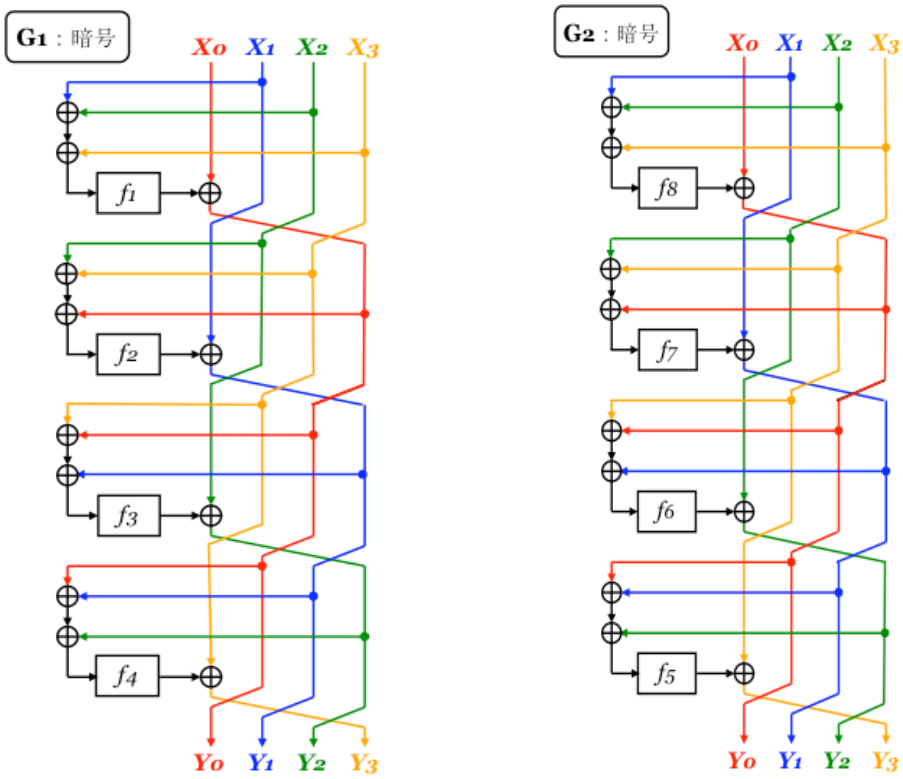
戻り値

なし

※コンパイル・オプションによっては、マクロ関数になる。

4.5 G関数

暗号化処理の一部機能



4.5.1 G1関数

構文

void G1(unsigned long* XtoY)

引数

XtoY [in/out] データをf関数を利用して暗号化して戻す［128bit］

戻り値

なし

※コンパイル・オプションによっては、マクロ関数になる。

4.5.2 G2関数

構文

void G2(unsigned long* XtoY)

引数

XtoY [in/out] データをf関数を利用して暗号化して戻す［128bit］

戻り値

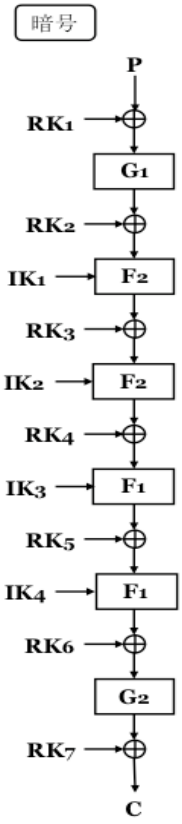
なし

※コンパイル・オプションによっては、マクロ関数になる。

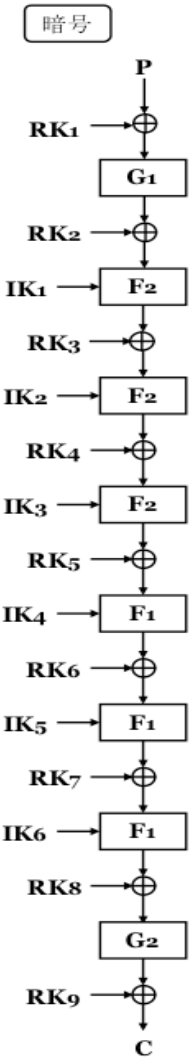
4.6 Hyral関数

HyRAL暗号化のメインモジュール
HyRALSingle:128bitKey暗号機能、HyRALDouble:192/256bitKey暗号機能

HyRAL Single:128bit Key Mode



HyRAL Double:192/256bit Key Mode



4.6.1 HyralSingle関数

HyRAL 128bitKey暗号のメインモジュール

構文
void HyralSingle(unsigned long* PtoC)

引数
PtoC [in/out] 入力データを暗号化して戻す[128bit]

戻り値
なし

4.6.2 HyralDouble関数

HyRAL 192/256bitKey暗号のメインモジュール

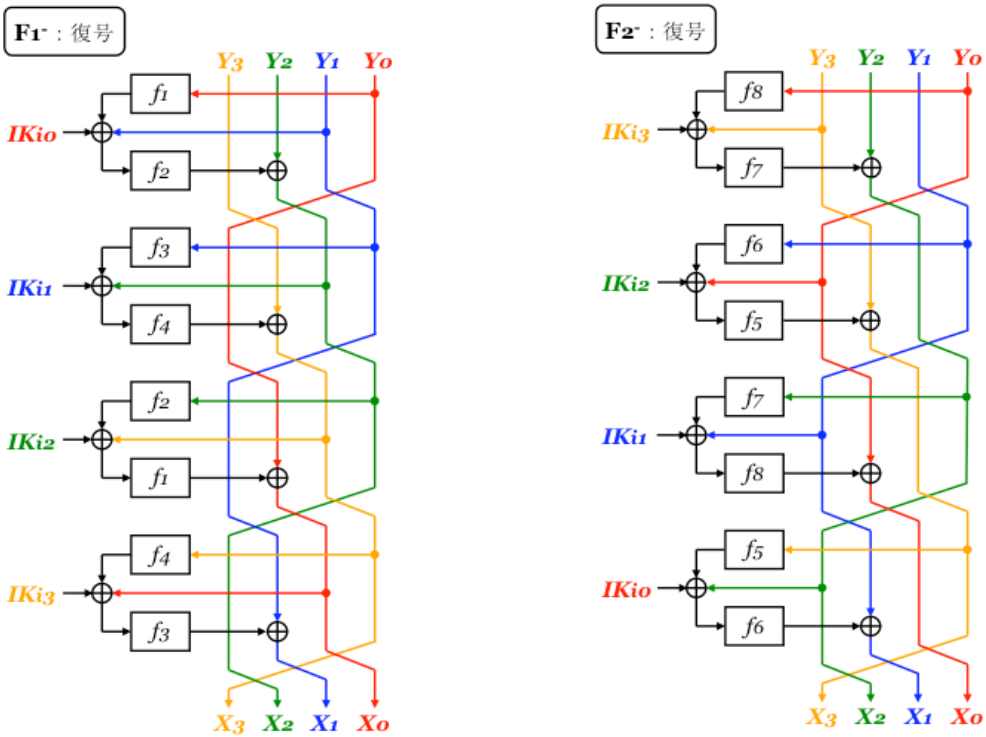
構文
void HyralDouble(unsigned long* PtoC)

引数
PtoC [in/out] 入力データを暗号化して戻す[128bit]

戻り値
なし

4.7 rF関数

復号化処理の一部機能



4.7.1　rF1関数

構文

```
void rF1(unsigned long* YtoX, unsigned long* IKi)
```

引数

YtoX	[in/out]	データをf関数とSubKeyを利用して復号化して戻す［128bit］
IKi	[in]	SubKey

戻り値

なし

※コンパイル・オプションによっては、マクロ関数になる。

4.7.2　rF2関数

構文

```
void rF2(unsigned long* YtoX, unsigned long* IKi)
```

引数

YtoX	[in/out]	データをf関数とSubKeyを利用して復号化して戻す［128bit］
IKi	[in]	SubKey

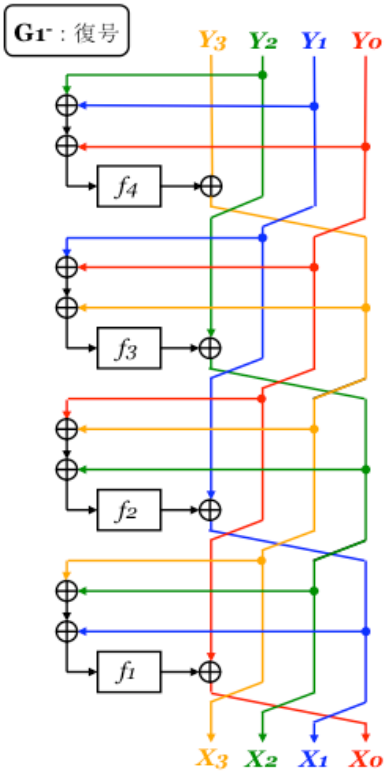
戻り値

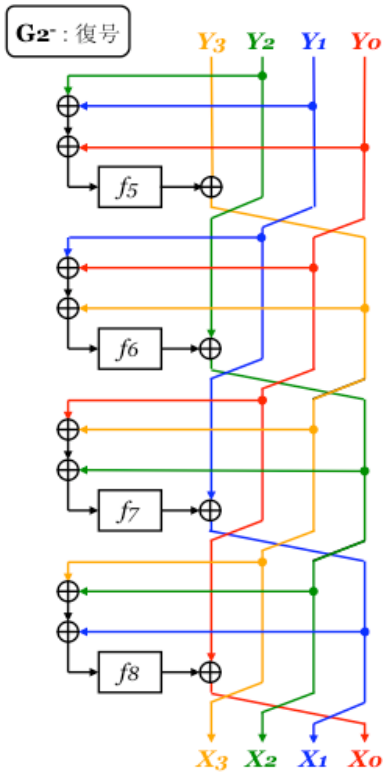
なし

※コンパイル・オプションによっては、マクロ関数になる。

4.8 rG関数

復号化処理の一部機能





4.8.1　rG1関数

構文

void rG1(unsigned long* YtoX)

引数

XtoY　　[in/out]　データをf関数を利用して復号化して戻す［128bit］

戻り値

なし

※コンパイル・オプションによっては、マクロ関数になる。

4.8.2　rG2関数

構文

void rG2(unsigned long* YtoX)

引数

XtoY　　[in/out]　データをf関数を利用して復号化して戻す［128bit］

戻り値

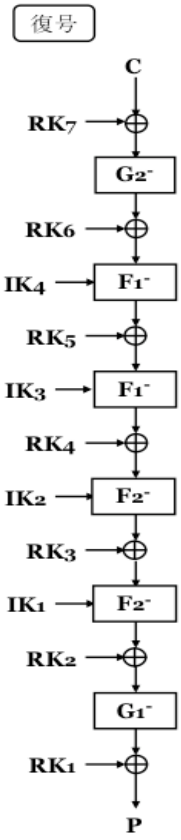
なし

※コンパイル・オプションによっては、マクロ関数になる。

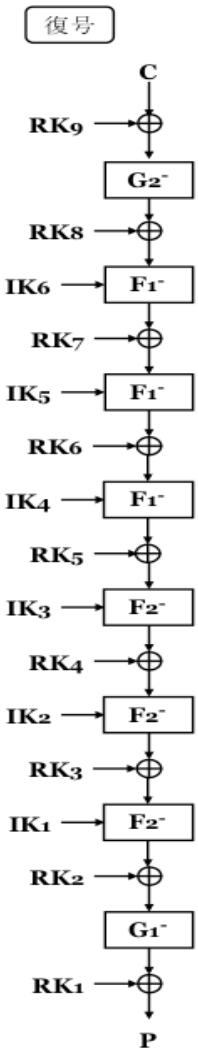
4.9 rHyral関数

rHyRAL復号化のメインモジュール
rHyRALSingle:128bitKey復号機能、rHyRALDouble:192/256bitKey復号機能

HyRAL Single:128bit Key Mode



HyRAL Double:192/256bit Key Mode



4.9.1 rHyralSingle関数

HyRAL 128bitKey復号のメインモジュール

構文
void rHyralSingle(unsigned long* CtoP)

引数
CtoP [in/out] 入力データを復号化して戻す[128bit]

戻り値
なし

4.9.2 rHyralDouble関数

HyRAL 192/256bitKey復号のメインモジュール

構文
void rHyralDouble(unsigned long* CtoP)

引数
CtoP [in/out] 入力データを復号化して戻す[128bit]

戻り値
なし