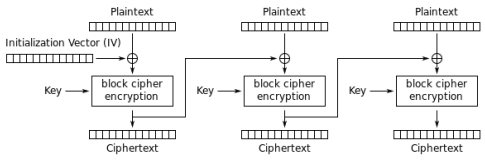
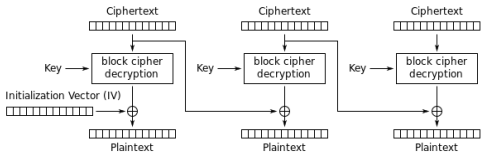
Simon Speck Rectangle

CBC

- CBC mode 计算规则
  1) 包括密钥编排、加密、解密三部分；
  2) RAM：明文、主密钥、轮子密钥、初始向量、解密辅助数据（保存临时密文），密钥编排辅助数据；
  3) Flash(data)
     a) 数据：不包括明文、主密钥、初始向量 vector，直接对 RAM 中的数据进行初始化；包括轮常量（比如 Simon 中的 Z、Rectangle 中密钥编排的常量）；
  4) Flash(code) & Time
     a) 密钥编排
        - 起始状态：主密钥初始化完成，轮常量初始化完成；
        - 结束状态：主密钥中数据不变，轮子密钥全部计算完成并保持在 RAM 中；
        - 包括内容：~~主密钥复制到轮密钥前 16bytes（以 Simon64/128 为例）~~、轮函数计算，程序框架（寄存器初始化，循环控制，子程序调用与返回，辅助数据更新）；
     b) 加密
        - 起始状态：明文、初始向量初始化完成，轮子密钥计算完成；
        - 结束状态：所有明文被密文覆盖、RAM 中初始向量和轮子密钥保存不变；
        - 包括内容：加载明文、加载初始向量、异或向量、轮函数加密、写回密文同时更新寄存器中向量、程序框架（寄存器初始化，不同 Block 间循环控制、同一 Block 内轮数循环控制，子程序调用与返回）；
     c) 解密
        - 起始状态：明文被密文覆盖，初始向量、轮子密钥没被修改；
        - 结束状态：密文被恢复成明文、RAM 中初始向量和轮子密钥保存不变；
        - 包括内容：加载密文、加载初始向量、密文复制（解密过程中密文会被覆盖，而在下一个 Block 解密时需要用到本次的密文，因此需要在密文被明文覆盖前先复制）、轮函数解密、异或向量、写回明文、使用复制的密文更新寄存器中向量、程序框架（寄存器初始化，不同 Block 间循环控制、同一 Block 内轮数循环控制，子程序调用与返回）；



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

			Simon	Speck	Rectangle
RAM	总计		336 bytes	384 bytes	360 bytes
	向量		8	8	8
	明文/密文		128	128	128
	主密钥		16	16	16
	轮子密钥		176	108	208
	密钥编排辅助数据		—	116	—
	保存密文		8	8	—
Flash (data)	总计		8 bytes	0	26 bytes
	轮常量		8	–	26(25 对齐)
Flash (code) Time	三部分总计 Flash(code)/Time		550 bytes 64910 cycles	544 bytes 42572 cycles	670 bytes 63901 cycles
	密钥编排	总计	162/3450	148/1846	158/1462
		装载主密钥	–	40/118	32/32
		存储主密钥	8/320	8/208	32/416
		S 盒	–	–	28/350
		Feistel	–	–	36/450
		轮常量异或	–	–	4/100
		密钥编排函数	112/2760	64/1040	–
		更新 L	–	12/312	–
		程序框架	40/370[1]	24/167[2]	26/114[3]
	加密	加密总计	178/30459	176/18603	254/31019
		加载向量	16/16	16/16	16/256
		向量异或	16/128	16/128	16/128
		加载明文	16/256	16/256	16/256
		S 盒	–	–	52/10400
		行移位	–	–	40/8000
		轮密钥异或	–	–	32/9600
		最后一轮密钥加	–	–	32/384
		明文加密	74/29552	72/17712	–
		更新向量	8/64	8/64	–
		写回密文	16/256	16/256	16/256
		程序框架	32/187	32/187	34/1707
	解密	解密总计	210/30971	220/22123	258/31420
		加载向量	16/16	16/16	16/256
		加载密文	16/256	16/256	16/256
		复制密文	16/256	16/256	—
		逆 S 盒	—	—	54/10800
		逆行移位	—	—	40/8000

批注 [a1]: 可以只使用 12 bytes 的 RAM 保存 L，编排过程中用后续的 L 覆盖最旧的，这样加密过程中对 L 的更新操作用时 1924cycles，整个密钥编排时间为 3508 cycles，代码量为 160 bytes。总消耗：
RAM：280bytes
Flash(code)：560bytes
Time：44264cycles

批注 [a2]: 4 条指令
8 bytes = 4 * 2
208 cycles = 8 * 26

批注 [a3]: 56 条指令
112 bytes = 56 * 2
2760 cycles = (69 * 40)
69：编排一轮需要 cycles
40：密钥编排轮数

批注 [a4]: 32 条指令
64 bytes = 32 * 2
1040 cycles = 40 * 26

批注 [a5]: 6 条指令
12 bytes = 6 * 2
312 cycles = 12 * 26

批注 [a6]:
29552 cycles = [(38+2)*44+2*43+1] * 16

批注 [a7]: 实现很巧妙，向量更新包括在里面

批注 [a8]:
32 = (5 + 4 + 5 + 1 + 1) * 2
187 = 5*1 + 4*16 +
(4*16+1*15+2*1+2*15) + 3 + 4

批注 [a9]: Simon、Speck 每一轮之间的循环控制算在了加密的部分，不是算在程序框架中；Rectangle 将这部分时间算在了程序框架中。
虽然稍有不同，但是都包括了。

		轮密钥异或	—	—	32/9600
		最后一轮密钥异或	—	—	32/384
		解密	74/29552	86/20720	—
		向量异或	16/128	16/128	16/128
		更新向量	16/256	16/256	—
		写回明文	16/256	16/256	16/256
		程序框架	40/251	38/235	36/1708

批注 [a10]:
40 = (5 + 6 + 7 + 1 + 1) * 2
251 = 5*1 + 6*16 +
(6*16+1*15+2*1+2*15) + 3 + 4

批注 [a11]: 和加密一样，每轮之间的循环控制时间计算在这部分

[1] Simon 密钥编排程序框架

40 = ((4+1+2) + (1+2+1) + 1 + 1 + (1+1+1) + 1 + 1 + 1 + 1) * 2

370 = (4+3+2)*1 + (2+2+1)*40 + (5*2+35) + (35*2) + (1+3+1)*5 + (1*2+4+2*4) + 3 + 4

- 4 条指令获取地址（4cycles），1 条指令加载 Z（3cycles），2 条指令初始化寄存器（2cycles）：各执行一次；
- 1 条地址变化指令（2cycles），2 条寄存器变化指令（2cycles），1 条比较指令（1cycles）：各执行 40 次；
- 一条 breq 指令，跳转 5 次（5*2cycles），另外 35 次不跳转（35cycles）；
- 一条 rjmp 指令执行 35 次（35*2cycles）；
- 一条寄存器清零指令（1cycles），1 条加载 Z 指令（3cycles），一条比较指令（1cycles）：各执行 5 次；
- 一条 breq 指令跳转 1 次（1*2cycles），另外 4 次不跳转（4cycles），一条 rjmp 指令执行 4 次（2*4cycles）；
- 1 条子程序调用指令（3cycles），1 条返回指令（4cycles）；

[2] Speck 密钥编排程序框架

24 = ((2 + 2) + (2 + 1 + 1 + 1) + 1 + 1 + 1) * 2

167 = (2+2)*1 + 2 + (2+1+1)*26 + 25*2 + 3 + 4

- 2 条指令获取地址（2cycles），2 条指令初始化寄存器（2cycles）：各执行一次；
- 2 条获取 L 地址指令（2cycles）；
- 1 条 sbiw 指令（2cycles），1 条轮数变化指令（1cycles），1 条比较指令（1cycles）：各执行 26 次；
- 1 条 brne 指令跳转 25 次（25*2cycles），另外 1 次不跳转（1cycles）；
- 1 条子程序调用指令（3cycles），1 条返回指令（4cycles）；

[3] Rectangle 密钥编排程序框架

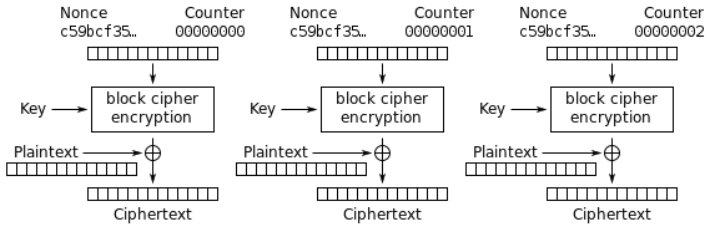
26 = (2 + 2 + 1 + 1 + 2 + 1 + 1 + 1 + 1 + 1) *2

114 = 2*1 + (2+1+1+2)*1 + 1*25 + (24 + 1*2 + 2*24) + 3 + 4

- 2 条指令获取主密钥 RAM 地址（2cycles）：各执行 1 次；
- 2 条指令获取轮子密钥 RAM 地址（2cycles），1 条指令设置总轮数寄存器（1cycles），1 条指令清零寄存器（1cycles），2 条指令获取轮常量 Flash 地址（2cycles）：各执行 1 次；
- 1 条指令轮数减（1cycles）：执行 25 次；
- 1 条指令判断是否是最后一轮：（执行 24 次不跳转，24*1cycles；1 次跳转，2cycles）；不是最后一轮时跳转到轮加密开始 1 条指令（2cycles，使用的是 rjmp 不是 jmp），执行 24 次；
- 1 条返回执行（4cycles），1 条子程序调用指令（3cycles）；

CTR：Low Flash 的实现（采用循环加密两个 Block）

- CTR mode 计算规则
  1) 只包括加密，没有密钥编排、解密部分；
  2) 没有 nonce，直接对计数器加密后与明文异或；
  3) RAM：明文，计数器；
  4) Flash(data)
     a) 数据：只包括轮子密钥，明文和计数器直接在 RAM 中进行初始化；
  5) Flash(code) & Time：加密（具体包括内容见下表）的代码和时间；



Counter (CTR) mode encryption

			Simon	Speck	Rectangle
RAM	总计		24 bytes	24 bytes	24 bytes
	明文/密文		16	16	16
	计数器		8	8	8
Flash (data)	总计		176 bytes	108 bytes	208 bytes
	轮常量		176	108	208
Flash (code) Time	总计 Flash(code)/Time		188 bytes 4181 cycles	186 bytes 2563 bytes	274 bytes 4295 cycles
	加密	加载计数器	16/16	16/16	16/16
		复制计数器	8/4	8/4	8/4
		计数器加 1	2/1	2/1	2/1
		加密计数器	74/4046	72/2428	–
		S 盒	–	–	52/1300
		行移位	–	–	40/1000
		轮密钥异或	–	–	32/1600
		最后一轮密钥加	–	–	32/64
		加载明文	16/32	16/32	16/32
		明文异或计数器	16/16	16/16	16/16
		第二次加载计数器	8/4	8/4	8/4
		写回密文	16/32	16/32	16/32
		程序框架	32/30[4]	32/30	36/226[5]

批注 [a12]: 37 条指令，两个 block
74 bytes = 37 * 2
4048 cycles = [(42+2)*44 + (43*2+1*1)] * 2

批注 [a13]: 36 条指令
72 bytes = 36 * 2
2428 cycles = [(41 + 2) *27 + (26*2+1)] * 2

批注 [a14]: 和 Simon CTR 程序框架一致

[4] Simon 加密程序框架

32 = ((2+2+1+1) + (2+1) + (1+1) + 1 + (1+1) + (1+1)) * 2

30 = (2+2+1+1) + (2+1)*2 + (1+1)*2 + (2+1) + 2+2+3+4

- 2 条指令获取计数器地址（2cycles），2 条指令获取明文地址（2cycles），1 条指令初始化块数（1cycles），1 条指令初始化零寄存器（1cycles）：各执行一次；
- 2 条指令获取轮子密钥地址（2cycles），1 条指令初始化轮数（1cycles）：各执行 2 次；
- 1 条轮数变化指令（1cycles），1 条比较指令（1cycles）：各执行 2 次；
- 1 条 brne 指令跳转 1 次（2cycles）、另一次不跳转（1cycles）；
- 1 条 adiw 指令执行 1 次（2cycles），1 条 rjmp 执行 1 次（2cycles）；
- 1 条子程序调用指令（3cycles），1 条返回指令（4cycles）；

[5]Rectangle 加密程序框架

36 = ((1+2+2+1) + (1+2) + 1 + (1+1) + (1+1+1+1) + (1+1)) * 2

226 = (1+2+2+1) + (2+1)*2 + 1*50 + (48*1+2*2+48*2) + 1*2 + (2+1+2+2) +3+4

- 1 条指令清零寄存器（1cycles），2 条指令获取计数器 RAM 地址（2cycles），2 条指令获取明文 RAM 地址（2cycles），1 条指令获取分组个数（1cycles）；
- 1 条指令获取总轮数（1cycles），2 条指令获取轮子密钥地址（2cycles）：各执行两次；
- 1 条指令轮数减（1cycles）：执行 50(25*2) 次；
- 1 条指令判断是否是最后一轮:（执行 48 次不跳转，48*1cycles；2 次跳转，2*2cycles）；1 条指令在不是最后一轮时跳转到轮加密开始（2cycles，使用的是 rjmp 不是 jmp），执行 48 次；
- 1 条指令分组数减一（1cycles），执行 2 次；
- 1 条指令判断是否为最后一个分组（1 次不跳转，1cycle；另 1 次跳转，2cycles），1 条指令在不是最后一个分组时跳到处理一个分组的开始（2cycles），1 条指令在是最后一个分组跳到最后终止（2cycles）；
- 1 条返回执行（4cycles），1 条子程序调用指令（3cycles）；