基于HASH函数的安全密钥扩展算法的研究

核探测与核电子学国家重点实验室
中国科学技术大学 近代物理系

导师：王坚
罗春丽

研究背景

SHA-256算法

SHA用于密钥扩展

研究背景

- 量子密钥分发简介
- QKD系统中密钥产生速率的状况
- 视频等通信对密钥的需求

SHA算法介绍

SHA家族算法由美国国家安全局（NSA）所设计，并由美国国家标准与技术研究院(NIST)发布

至今未出现对SHA-2的有效碰撞算法，从运算复杂度、运算速度等方面综合考虑，选用SHA-256

SHA-256的输入为1-2^{64}bit，输出固定长度256bit

算法分为两个部分：

1.预处理

2.计算

预处理

序列 M：L bits，在M的后面添加bit“1”，
再添加k个0，再添加值为L的64bit块，使得
(L+1+k+64)% 512 = 0 ，(L+1+k+64=N*512)
N=1
初始化8个Hash值：Ho..... H7

计算

从预处理得到的512bits数据块，得到64个Wt

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

初始化8个变量：a=H0；b=H1；……g=H7

六种基本函数

- $Ch(x,y,z)=(x \wedge y) \oplus (-x \wedge z)$
- $Maj(x,y,z)(z \wedge) \oplus (x \wedge z) \oplus (x \wedge z)$
- $\sum^{\{256\}}_{0}(x)= ROTR^{2}(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$
- $\sum^{\{256\}}_{1}(x)= ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$
- $\sigma^{\{256\}}_{0}(x)= ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x)$
- $\sigma^{\{256\}}_{1}(x)= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$

- 执行64次后，得到新的Hash值

$$T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$

$$T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

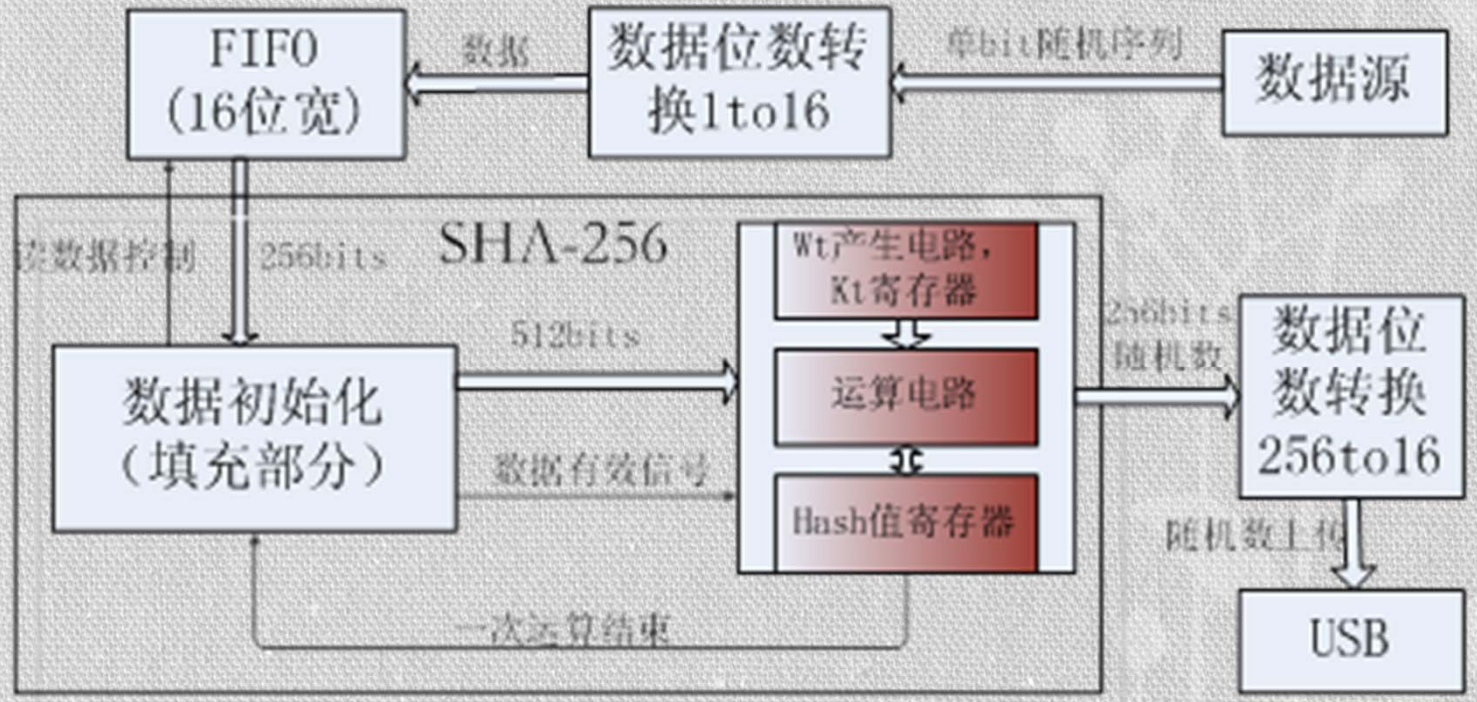
$$a = T_1 + T_2$$

$$H_0^{(i)} = a + H_0^{(i-1)}$$

……

$$H_7^{(i)} = h + H_7^{(i-1)}$$

FPGA中实现框图


FIFO
(16位宽)
数据
数据位数转
换1to16
单bit随机序列
数据源
SHA-256
256bits
Wt产生电路，
Kt寄存器
512bits
数据初始化
（填充部分）
运算电路
256bits
随机数
数据位
数转换
256to16
数据有效信号
Hash值寄存器
随机数上传
一次运算结束
USB

扩展密钥测试

采用美国国家标准和技术研究所（NIST）提供的随机数测试程序

小倍率算法的2倍、4倍、8倍以及大倍率算法的8倍、16倍、32倍均通过了NIST的所有测试项

更大扩展倍数则不能完全通过测试

根据SHA-256提出两种密钥扩展算法

小倍率算法

根据标准的SHA-256算法，输入小于256bits，输出256bits

大倍率算法

64次迭代的中间Hash值与最终Hash值同时作为扩展密钥输出

小倍率算法测试结果

倍率 / 测试项	小倍率算法(2倍)	小倍率算法(4倍)	小倍率算法(8倍)
Frequence	0.546283/0.988	0.095713/0.994	0.092041/0.987
BlockFrequency	0.473064/0.990	0.203351/0.991	0.463512/0.987
CumulativeSums*	0.253733/0.987	0.558482/0.993	0.073411/0.988
Runs	0.639202/0.993	0.886905/0.991	0.972382/0.986
LongestRun	0.296834/0.985	0.361938/0.991	0.088226/0.992
Rank	0.672470/0.993	0.782063/0.991	0.410055/0.989
FFT	0.093720/0.987	0.023545/0.988	0.083018/0.984
NonOverlappingTemplate*	0.506294/0.989971	0.502984/0.99	0.508395/0.99
OverlappingTemplate	0.508172/0.987	0.564639/0.99	0.556460/0.987
Universal	0.884671/0.992	0.164882/0.988	0.108150/0.995
ApproximateEntropy	0.870856/0.99	0.914025/0.992	0.641284/0.992
RandomExcursions*	0.580435/0.988	0.794475/0.989	0.452195/0.989
RandomExcursionsVariant*	0.453539/0.988	0.472389/0.991	0.626995/0.994
Serial*	0.698263/0.988	0.685302/0.992	0.626723/0.992
LinearComplexity	0.530120/0.992	0.967382/0.989	0.357000/0.991

大倍率算法测试结果

测试项 \ 倍率(输入64bits)	8倍[64+last]	8倍[56+last]	16倍[48+56+64+last]	32倍
Frequence	0. 150552/0.993	0.786355 /0.989	0.153763 /0.991	0.490483/0.991
BlockFrequency	0. 650132/0.990	0.603322/0.985	0.952152 /0.989	0.686955/0.993
CumulativeSums*	0. 494509/0.993	0.909354/0.988	0.724871/0.992	0.39048 /0.990
Runs	0. 180322/0.994	0.324821 /0.991	0.709558 /0.988	0.092877/0.990
LongestRun	0. 524101/0.985	0.754832 /0.986	0.454053 /0.987	0.389855/0.989
Rank	0.696743/0.995	0.410055 /0.99	0.365253 /0.987	0.613187/0.99
FFT	0.444226/0.985	0.315297 /0.984	0.715679 /0.989	0.004962/0.989
NonOverlappingTemplate*	0.468880 /0.99	0.512353 /0.99	0.475415 /0.990	0.527472/0.990
OverlappingTemplate	0.388127 /0.99	0.187836 /0.99	0.358641 /0.983	0.335324/0.989
Universal	0.270603/0.988	0.754832 /0.993	0.320607 /0.988	0.104682/0.985
ApproximateEntropy	0.451234 /0.981	0.827722/0.985	0.152902/0.986	0.248649/0.987
RandomExcursions*	0.554589 /0.989	0.302469 /0.991	0.59169 /0.986	0.485055/0.988
RandomExcursionsVariant*	0.47965 /0.991	0.396458 /0.989	0.398799/0.991	0.449292/0.990
Serial*	0.620341/0.99	0.737975 /0.995	0.405068 /0.99	0.127318/0.990
LinearComplexity	0.474986/0.989	0.871642/0.99	0.61007/0.993	0.360287/0.992

谢 谢 ！