密 码 学

第十章 密码学的新方向

网络空间安全学院
胡 伟　朱丹
weihu/zhudan@nwpu.edu.cn

知识回顾-RSA算法参数的选取

- (1) p和q要足够大：p和q应至少为512位，使n达1024位
- (2) p和q要是强素数：(p-1)、(p+1)、(q-1)、(q+1)均有大素数因子
- (3) p和q位数相差不能太小，也不能太大
- (4) 在给定的条件下，找到的*d* = *e*，这样的密钥必须舍弃
- (5) 如果使得$e^i = 1 \bmod \varphi(n)$成立的i值很小，则很容易进行密文迭代攻击
- (6)(7) *e*和*d*的选择：兼顾安全性和实现效率

https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

知识回顾-RSA算法参数的选取

- (8) 模数n的使用限制
  - 不要许多用户共用一个模n，否则易受共模攻击
  - 设用户A的加密密钥为e_A，用户B的加密密钥为e_B，他们使用同一个模数n，对于同一条明文有

$$C_A = M^{e_A} \bmod n$$

$$C_B = M^{e_B} \bmod n$$

  - 当e_A和e_B互素时，可利用欧几里德算法求出两个整数r和s，使得

$$re_A + se_B = 1$$

  - 于是，${C_A}^r {C_B}^s = M^{re_A + se_B} = M \bmod n$

知识回顾-大素数的产生

- 概率产生法
  - 目前最常用的概率性算法是Miller检验算法
  - Miller检验算法已经成为美国的国家标准

- Miller检验算法
  - 欧拉定理：若n, a为正整数，且n, a互素，则有$a^{\varphi(n)} \equiv 1 \bmod n$
  - 费马小定理（欧拉定理的特殊情况）：如果p是一个素数，而整数a不是p的倍数（a和p互素），则有$a^{p-1} \equiv 1 \bmod p$
  - 不断取$a \in [1，p - 1]$，且$a \in Z$，验证$a^{p-1} \equiv 1 \bmod p$是否成立，不成立则p肯定不是素数，共取s次
  - 若s次均通过测试，则p不是素数的概率不超过2^{-s}

知识回顾-大素数的产生

- Miller检验算法是检测*p*为素数的必要条件1
- 必要条件2：如果p为素数，则方程

$$x^2 \equiv 1 \bmod p$$

只有x = 1和x = -1（ x = p - 1）两个解。若方程存在其它解，则*p*不是素数

- 测试方法：给定奇数*p*，则存在q和m使得

$$p \ -1 = 2^q m,\ q \geq 1$$

- 构造序列$a^m \bmod p$， $a^{2m} \bmod p$， $a^{4m} \bmod p$， $\cdots$， $a^{2^q m} \bmod p$，正好最后一项是$a^{p-1} \bmod p$，并且后一项都是前一项的平方
- 若*p*为素数，则有，最后一项$a^{p-1} = 1 \bmod p$，且对于序列其中任何为1的项，其前项为1或*p* - 1

知识回顾-RSA密钥产生及加解密

- 加解密运算
  - 加密运算： $C = M^e \bmod n$
  - 解密运算： $M = C^d \bmod n$

- 模幂运算算法
  - 模幂运算基本定义
  - 重复平方算法
  - 滑动窗口算法
  - CRT算法
  - 蒙哥马利算法（了解）

知识回顾-重复平方算法

- 从右至左：$M = C^d \bmod n$

$$d = (d_{w-1}, d_{w-2}, \ldots, d_1, d_0)$$

$$d = d_{w-1}*2^{w-1} + d_{w-2}*2^{w-2} + \ldots + 2^1*d_1 + 2^0*d_0$$

$$M = C^d \bmod n = C^{d_{w-1}*2^{w-1}+d_{w-2}*2^{w-2}+\cdots+d_1*2^1+d_0}$$

$$= (C^{d_{w-1}})^{2^{w-1}} * (C^{d_{w-2}})^{2^{w-2}} * \cdots (C^{d_1})^2 * C^{d_0}$$

$C^{2\wedge(w-1)}$ / 1 $\quad$ $C^{2\wedge(w-2)}$ / 1 $\quad$ … $\quad$ C^2 / 1 $\quad$ C / 1

- 通过逐次平方，依次计算C^2，C^4，…，$C^{2^{w-2}}$，$C^{2^{w-1}}$
- 当$d_i = 1\ (0 \le i \le w-1)$时，结果乘以$C^{2^i}$

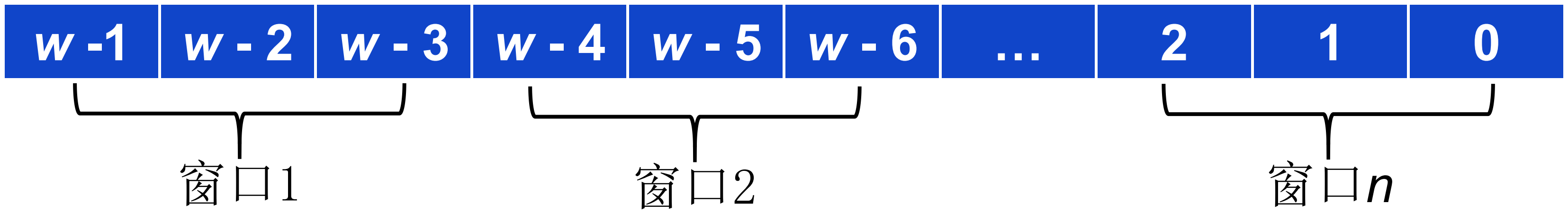
$$C^{2^i} = C^{2^{i-1}*2} = (C^{2^{i-1}})^2$$

- 当$d_i = 0$时，$C^{d_i} = 1$，乘以 1 (即1^{2^i})

知识回顾-滑动窗口算法

Sliding Window

- 每次处理一个窗口大小(*k*比特密钥)
- 分为固定窗口和可变窗口
- 分为从左至右和从右至左
- 重复平方法是窗口大小为1的特例
- 滑动窗口算法是从二进制数到多进制数情况的扩展
- 例如，窗口大小为3时，即为8进制数的情况



知识回顾-CRT算法

- 中国剩余定理(Chinese Remainder Theorem, CRT)表明，如果，

$$a = x \bmod n$$

$$b = x \bmod m$$

- 那么x存在唯一mod mn的解，当且仅当$gcd(m, n) = 1$

- 对于RSA，m和n为素数p和q，计算$M \bmod p$ 和$M \bmod q$ ：

$$M_p = C^d \bmod p = C^{d \bmod p-1} \bmod p \qquad C^{\varphi(p)} = 1 \bmod p$$

$$M_q = C^d \bmod q = C^{d \bmod q-1} \bmod q \qquad C^{\varphi(q)} = 1 \bmod q$$

- 那么C^d存在唯一mod pq的解，当且仅当$gcd(p, q) = 1$

知识回顾-CRT算法

- 计算常数$T = p^{-1} \bmod q$，

$$u = (M_q - M_p) * T \bmod q$$

$$M = M_p + u * p$$

- 其中，$M_p = C^d \bmod p$ 和 $M_q = C^d \bmod q$：

$$M_p = C^d \bmod p = C^{d \bmod p-1} \bmod p$$

$$M_q = C^d \bmod q = C^{d \bmod q-1} \bmod q$$

基本思想：将mod n级别的运算转化为mod p和mod q级别的运算

知识回顾-蒙哥马利算法

蒙哥马利乘法

- 优化模乘$x \cdot y \ mod \ q$运算中取模环节
- 原始取模操作：除法取余
- 蒙哥马利算法：转化为移位
  - 以R表示约简操作（R 为 2 的幂）
  - 首先将变量转化为Montgomery形式，如x转化为$xR \bmod q$
  - $xR \cdot yR = zR^2$, 通过Montgomery约简得$zR^2 \cdot R^{-1} = zR$
  - 约简后的zR仍为Montgomery形式，可继续参与后续运算
  - 最终结果乘以R^{-1} mod q转回标准形式（非Montgomery）

章节安排

Outline

- RSA时间侧信道攻击
- RSA时间侧信道防护

学习目标

- 了解密码算法实现面临的侧信道安全威胁
- 理解RSA算法实现时间侧信道产生原理
- 掌握基本的RSA时间侧信道攻击方法
- 了解常用的RSA时间侧信道防御措施

章节安排

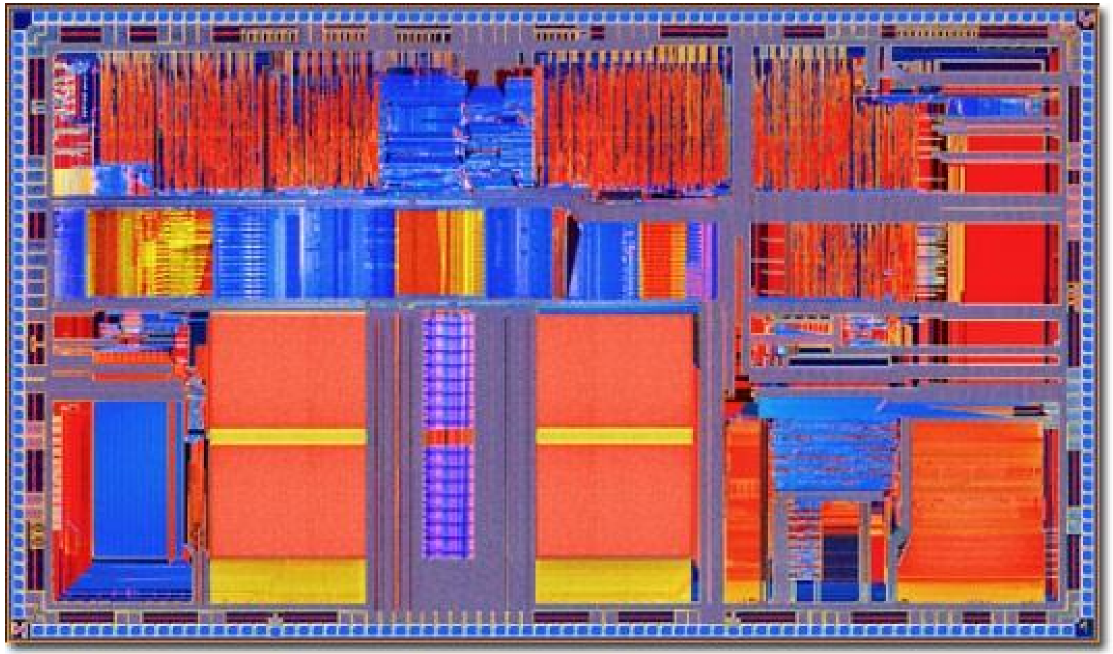
Outline

RSA时间侧信道攻击

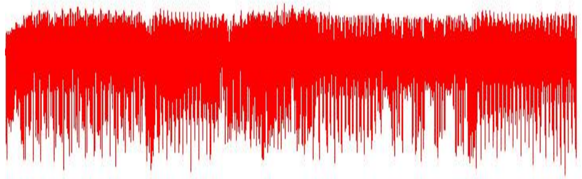
RSA时间侧信道防护

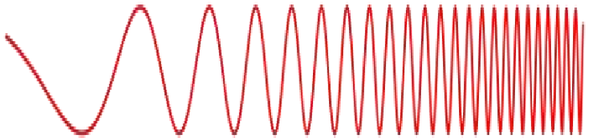
5.6(1) 侧信道信息泄露

5.6(1) 侧信道信息泄露

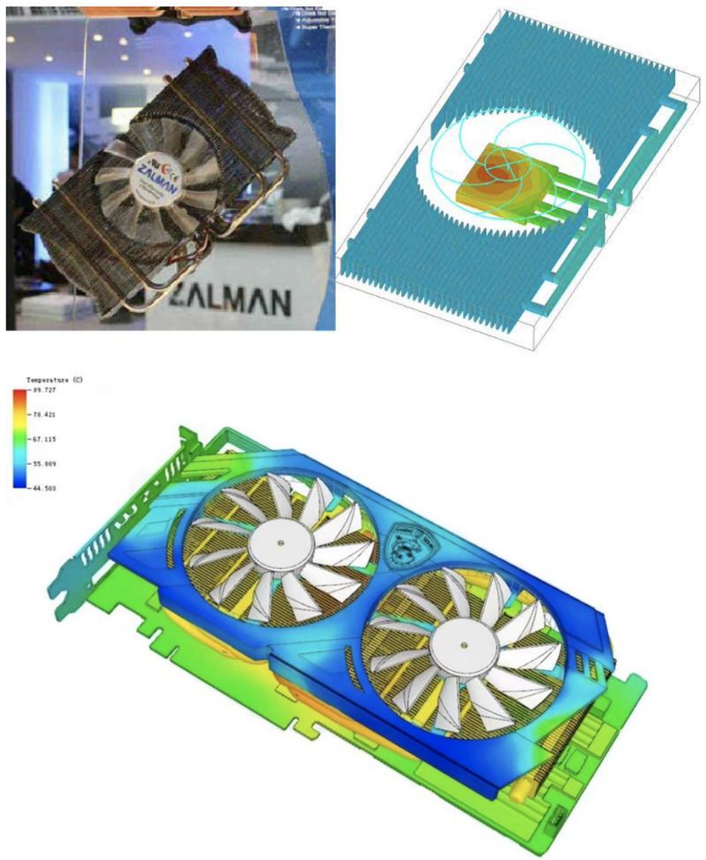



0101001010
0 010 101010
0101001001
0101001010
01 10100101
0101 010101
0101001010





5.6(1) 侧信道信息泄露



5.6(2) 侧信道类型

侧信道类型	基本原理	典型攻击对象
能量侧信道	利用密码算法芯片在不同密钥和明文输入条件下能量消耗的差异	AES、RSA
电磁侧信道	利用密码算法芯片在不同密钥和明文输入条件下电磁辐射强度的差异	AES、RSA
时间侧信道	利用密码算法芯片在不同密钥和明文输入条件下加密时间的差异	RSA、ECC
声学侧信道	利用密码算法芯片在不同密钥和明文输入条件下物理噪声的差异	RSA、键盘
光学侧信道	利用密码算法芯片在不同密钥和明文输入条件下光辐射的差异	AES、RSA

5.6(2) RSA时间侧信道产生原理

- 重复平方算法（从左至右）： $M = C^d \bmod n$

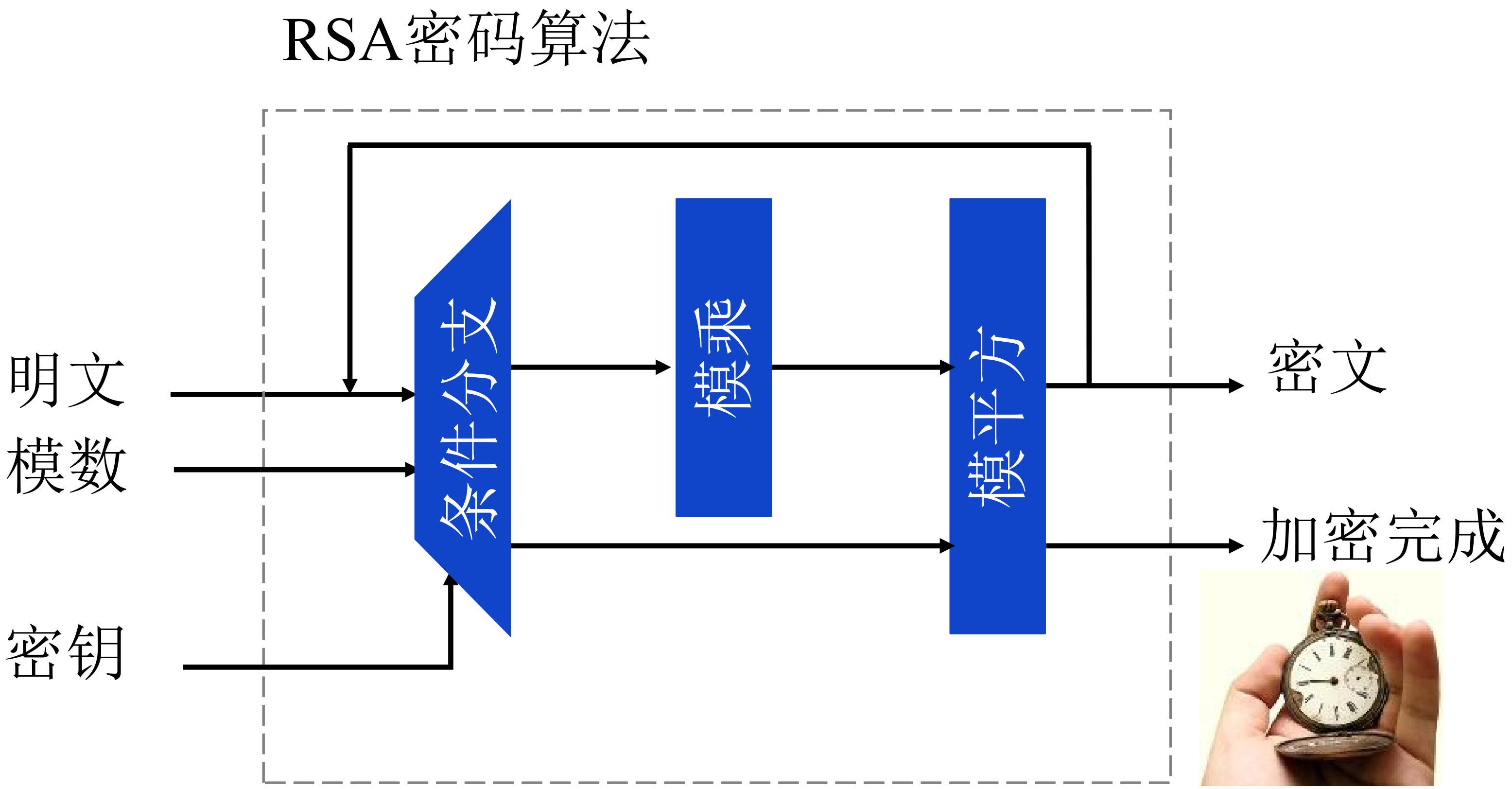
```
s[w] := 1
for i := w − 1 to 0 do
    if d[i] == 1 then
        m[i] := s[i + 1] * c mod n
    else
        m[i] := s[i + 1] * 1
    end if
    s[i] := m[i] * m[i] mod n
end for
return m[0]
```

5.6(2) RSA时间侧信道产生原理

- 重复平方算法（从右至左）： $M = C^d \bmod n$

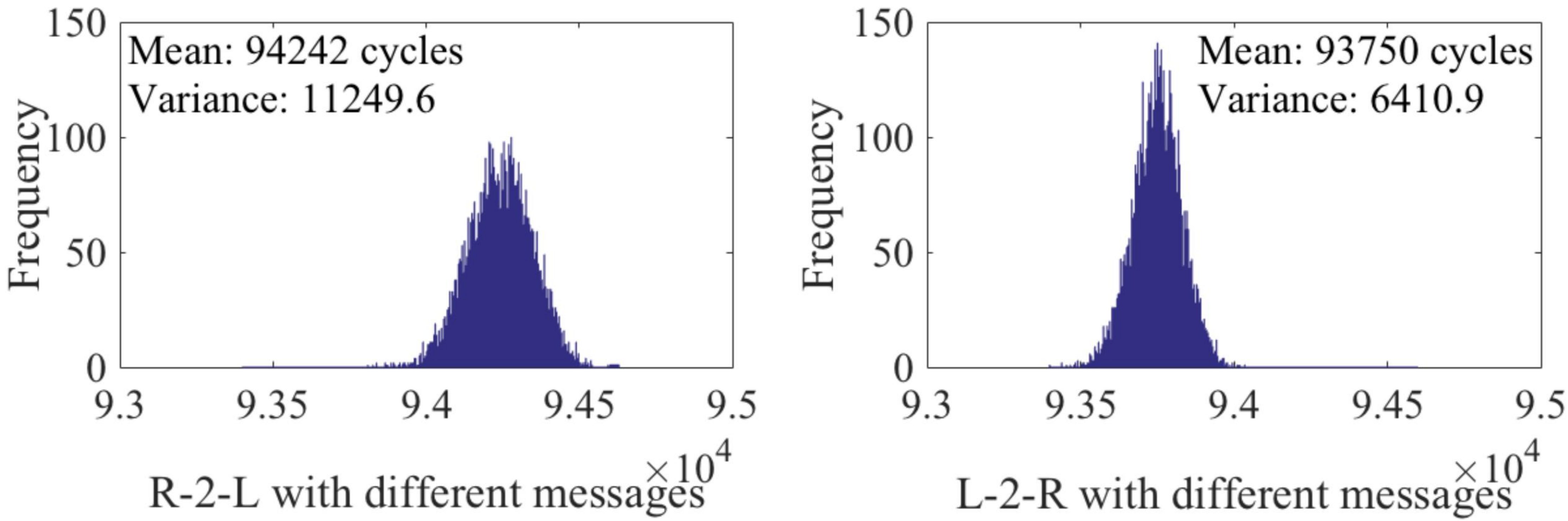
```
m[0]:=1
s[0]:=c
for i := 0 to w−1 do
    if d[i] == 1 then
        m[i+1] := m[i] * s[i] mod n
    else
        m[i + 1] := m[i] * 1
    end if
    s[i+1] := s[i] * s[i] mod n
end for
return m[w]
```

5.6(2) RSA时间侧信道产生原理



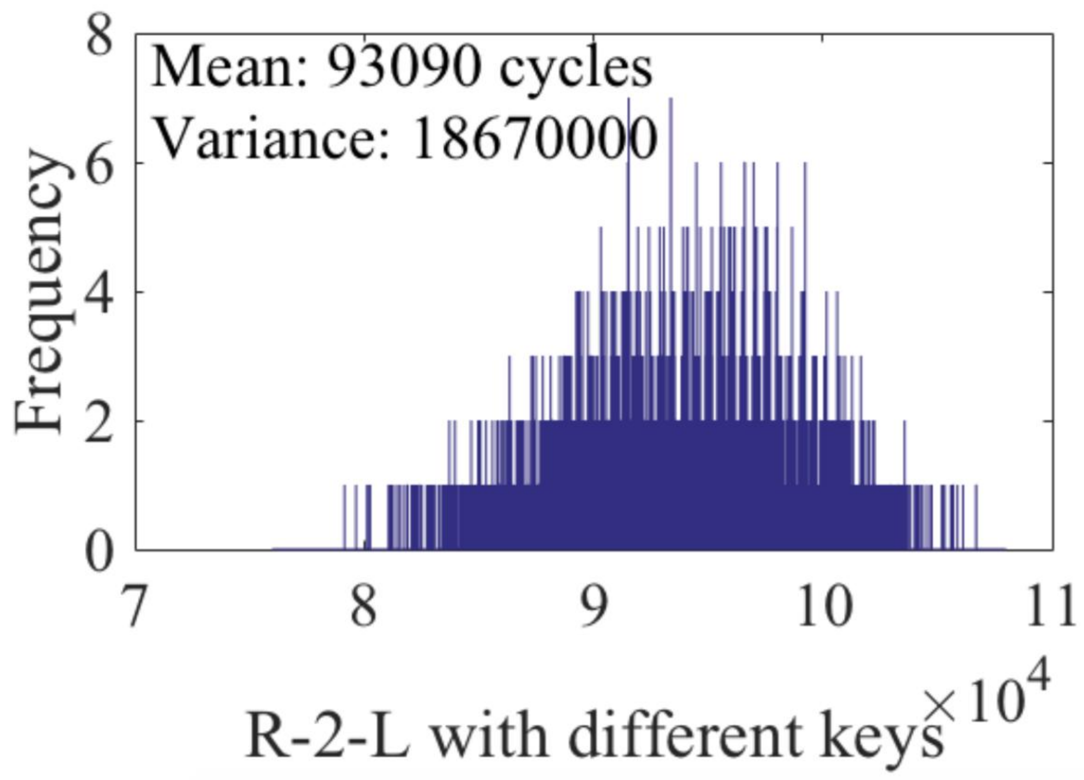
5.6(3) RSA密码算法加密时间分布

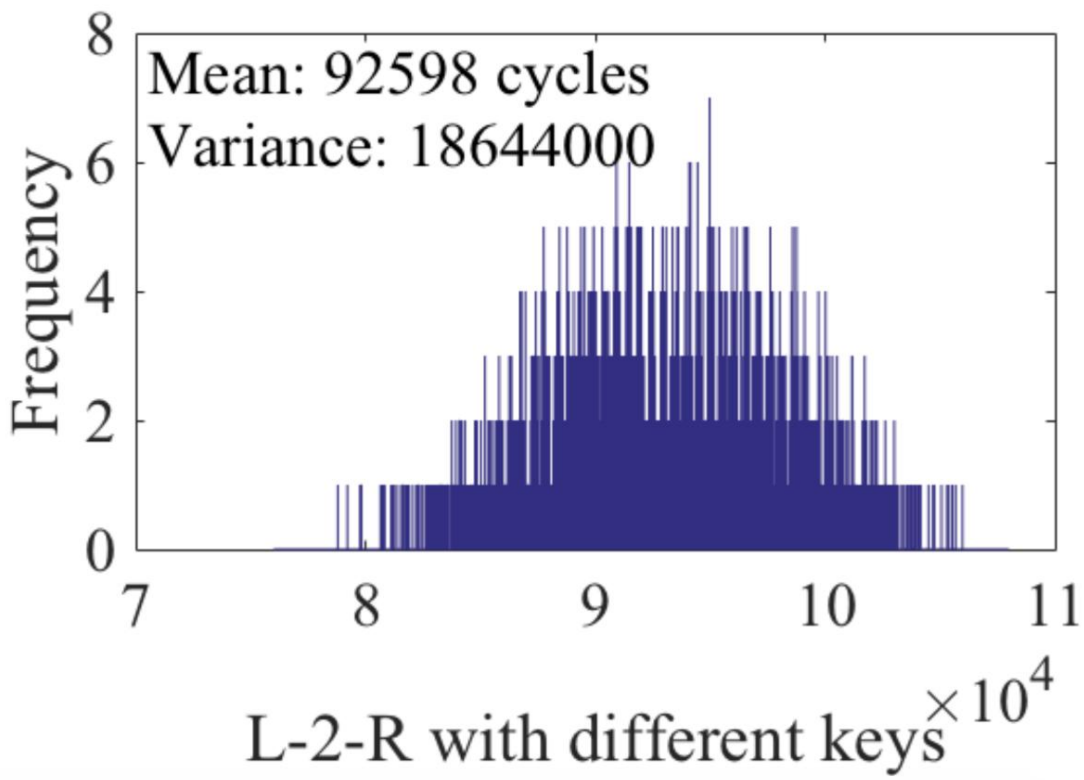
- 相同密钥和不同明文下加密时间的分布



5.6(3) RSA密码算法加密时间分布

- 相同明文和不同密钥下加密时间的分布





5.6(4) Kocher提出的时序攻击方法

- 1996年，美国科学家Paul Kocher博士发现密码芯片运算时泄漏的执行时间信息能被用于密码分析攻击，并成功果破译了Diffie-Hellman和RSA等密码协议和算法
- Kocher开启了密码侧信道分析时代

P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," Advances in Cryptology - CRYPTO'96, Springer-Verlag Lecture Notes in Computer Science, vol. 1109, pp. 104– 113, 1996.

https://www.paulkocher.com/index.html

5.6(5) RSA时序攻击数学基础

- 方差（Variance）
  - 方差是衡量随机变量或一组数据离散程度的度量
  - 概率论中方差用来度量随机变量和其数学期望（即均值）之间的偏离程度
  - 统计中的方差（样本方差）是每个样本值与全体样本值的数学期望之差的平方和的平均值
  - 设$x_1, x_2, \cdots, x_n$是随机变量X的一个样本，样本方差定义为

$$var(X) = \frac{\sum_{i=1}^{n}(x_i - \overline{X})^2}{n-1}$$

  - 其中，$\overline{X}$是 $x_1, x_2, \cdots, x_n$的样本均值
  - 若X和Y是两个独立的随机变量，则有

$$var(X \pm Y) = var(X) + var(Y)$$

5.6(5) RSA时序攻击数学基础

- 用T表示RSA算法加密N条明文的时间向量集合，其中$T_i(1 \le i \le N)$表示加密第i条明文所需的总时间
- 假设RSA密钥长度为M个比特，用$t_{i,j}(1 \le j \le M)$表示加密第i条明文时，处理第j个密钥位所需的时间
- 用$t_j(1 \le i \le M)$表示矩阵T中的M个列向量

$$
\begin{array}{ccc}
 & & \begin{array}{cccc}(t_1 & t_2 & \cdots & t_M)\end{array} \\
T = \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{pmatrix}_{N\times 1} & = & \begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,M} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ t_{N,1} & t_{N,2} & \cdots & t_{N,M} \end{pmatrix}_{N\times M}
\end{array}
\qquad T_i = \sum_{j=1}^{M} t_{i,j}
$$

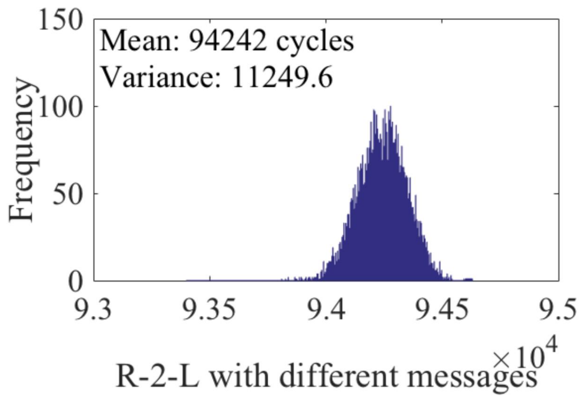
5.6(5) RSA时序攻击数学基础

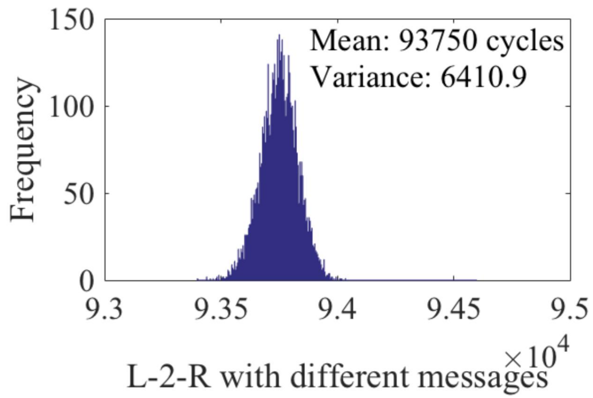
- 正态分布假设：多条明文下的加密时间T呈正态分布

- 独立性假设：假设RSA密码算法加密各个密钥位所需的时间是独立的，即某一个密钥位的处理时间不影响其它密钥位的处理时间

- 矩阵T中的各个列向量 $t_1, t_2, \cdots, t_M$是统计独立的，因此有

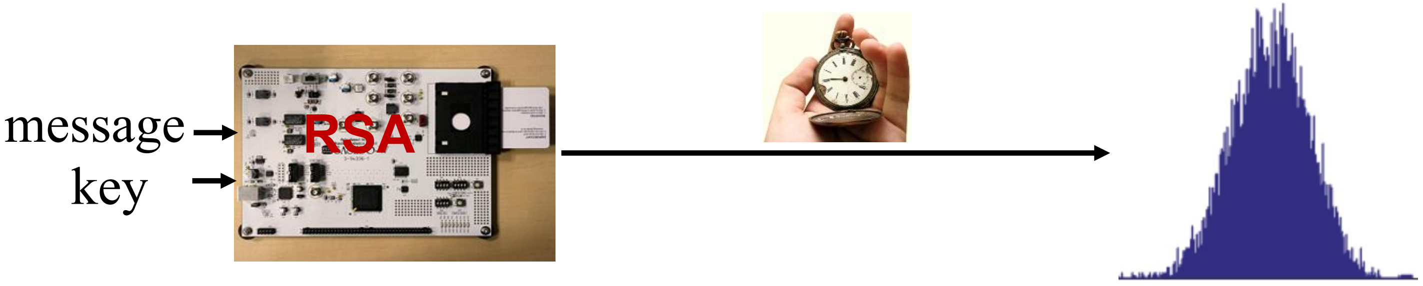
$$var(T) = var\left(\sum_{j=1}^{M} t_j\right) = \sum_{j=1}^{M} var(t_j)$$





5.6(6) RSA时序攻击方法

- 攻击模型
  - 攻击者知道设计细节
  - 攻击者可以给定明文，并观测加密时间
  - 攻击者可以猜测并尝试密钥



$$T = \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{pmatrix}_{N\times 1}$$

$$(t_1 \quad t_2 \quad \cdots \quad t_M)$$

5.6(6) RSA时序攻击方法

- 攻击步骤
  - S1：给定N条明文输入，观测相应的加密时间，获得向量T
  - S2：最低密钥位始终为1
  - S3：假设已经恢复了1到$i-1 (i \geq 2)$个密钥位
  - S4：从第$i (i \geq 2)$个密钥位开始，分别猜测其为0和1，其它高密钥位（大于i的密钥位）为先置为0
  - S5：在相同的N条明文输入下，采集第i个密钥位分别为0和1时的加密时间向量t_i^0和t_i^1
  - S6：分别计算$var(T-t_i^0)$和$var\left(T-t_i^1\right)$
  - S7：若$var(T-t_i^0) < var(T-t_i^1)$，则第i个密钥位为0，否则第i个密钥位为1

5.6(6) RSA时序攻击方法

- 基本思想
  - 给定N条明文输入，观测相应的加密时间，获得矩阵T
  - 分别猜测第i个密钥位为0或1，在相同的明文输入下，采集第i个密钥位分别为0和1时的加密时间向量t_i^0和t_i^1
  - 矩阵T中的各个列向量 $t_1, t_2, \cdots, t_M$是统计独立的
  - 如果密钥猜测c(c = 0或1)正确，则t_i^c是T的一部分，根据$var(T) = var\left(\sum_{j=1}^{M} t_j\right) = \sum_{j=1}^{M} var(t_j)$有，

$$var(T - t_i^c) = var(T) - var(t_i^c) < var(T)$$

  - 否则， t_i^c与T无关（统计独立），根据方差的性质有

$$var(T - t_i^c) = var(T) + var(t_i^c) > var(T)$$

5.6(6) RSA时序攻击方法

攻击实验数据示例

T	t1-0	t1-1	t2-0	t2-1	t3-0	t3-1	t4-0	t4-1	t5-0	t5-1	t6-0	t6-1	t7-0	t7-1	t8-0	t8-1	t9-0	t9-1
3248	8	135	135	258	258	393	393	528	393	573	573	704	704	839	839	970	970	1101
3290	8	127	127	262	262	385	385	516	385	587	587	718	718	849	849	980	980	1111
3256	8	123	123	250	250	377	377	512	377	581	581	712	712	839	839	962	962	1097
3264	8	135	135	262	262	393	393	524	393	595	595	730	730	865	865	1000	1000	1135
3264	8	139	139	270	270	393	393	520	393	589	589	716	716	839	839	966	966	1093
3286	8	143	143	266	266	401	401	536	401	605	605	740	740	871	871	1006	1006	1137
3270	8	135	135	262	262	397	397	532	397	573	573	708	708	835	835	970	970	1089
3300	8	135	135	266	266	397	397	520	397	595	595	722	722	857	857	992	992	1127
3318	8	139	139	274	274	401	401	532	401	603	603	738	738	873	873	1008	1008	1139
3244	8	131	131	266	266	389	389	524	389	593	593	720	720	851	851	986	986	1121
3266	8	139	139	266	266	389	389	524	389	585	585	716	716	843	843	978	978	1109
3274	8	143	143	278	278	413	413	544	413	603	603	726	726	849	849	984	984	1119
3264	8	139	139	270	270	401	401	524	401	595	595	730	730	857	857	992	992	1127
3252	8	143	143	278	278	373	373	508	373	573	573	704	704	831	831	962	962	1097
3234	8	135	135	270	270	401	401	536	401	605	605	704	704	839	839	974	974	1109
3226	8	127	127	258	258	389	389	520	389	591	591	710	710	825	825	960	960	1091
3226	8	143	143	274	274	409	409	544	409	597	597	712	712	847	847	958	958	1089
3266	8	143	143	270	270	397	397	528	397	595	595	722	722	857	857	992	992	1123
3188	8	139	139	270	270	393	393	528	393	597	597	732	732	867	867	1002	1002	1125
3280	8	143	143	270	270	393	393	528	393	585	585	720	720	851	851	986	986	1113

5.6(6) RSA时序攻击方法

- 32位RSA攻击结果分析示例（最终剩余方差52.44）

Init. & Bits 0-7	Var	793.6	761.3	759.7	725.2	725.2	685.1	649.2	646.0
	ΔVar	-	32.3	1.6	34.5	0.0	40.2	35.9	3.1
Bits 8-15	Var	600.6	597.7	567.4	535.0	495.0	495.0	451.1	413.3
	ΔVar	45.4	2.9	30.3	32.4	40.0	0.0	44.0	37.7
Bits 16-23	Var	413.3	365.8	365.8	361.2	322.7	291.4	257.1	224.0
	ΔVar	0.0	47.5	0.0	4.7	38.5	31.3	34.3	33.1
Bits 24-31	Var	223.7	189.9	154.8	117.2	86.49	83.60	52.44	52.44
	ΔVar	0.4	33.7	35.1	37.6	30.7	2.9	31.2	0.0

5.6(6) RSA时序攻击方法

- 利用方差（Variance）作为度量
- 将方差减小量作为密钥猜测正确性的判据

是否可采用其它度量?

5.6(6) RSA时序攻击方法

- 利用熵（Entropy）作为度量
  - 熵是随机变量不确定性的度量
  - 也可用于衡量随机变量或一组数据离散程度
  - 设$X = \{x_1, x_2, \ldots, x_n\}$是一个$n$元事件集合，$p$是集合$X$上的一个概率分布，即$x_i$出现的概率为$p(x_i) \geq 0$，且$\sum_{i=1}^{n} p(x_i) = 1$，则集合$X$中事件$x_i$出现时提供的信息量的数学期望

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) log_2\, p(x_i)$$

章节安排

Outline

RSA时间侧信道攻击

RSA时间侧信道防护

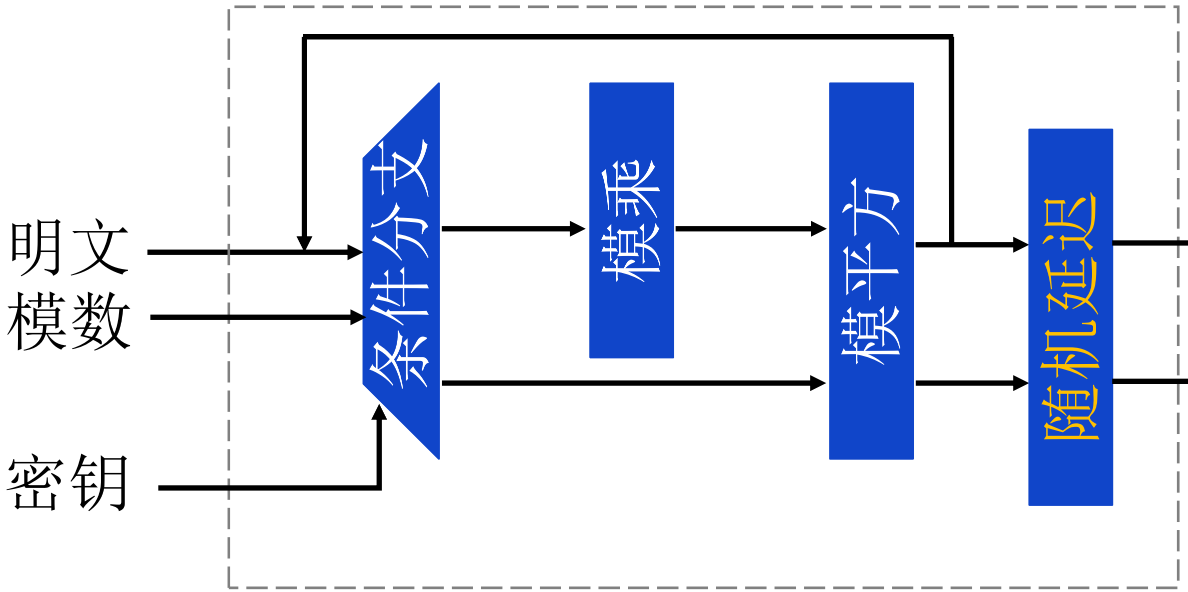
5.7(1) RSA时序防御方法

- 加随机延迟
- RSA Blinding
- 分支平衡化（空操作、Power Ladder）
- 蒙哥马利（Montgomery）乘法

- 加随机延迟

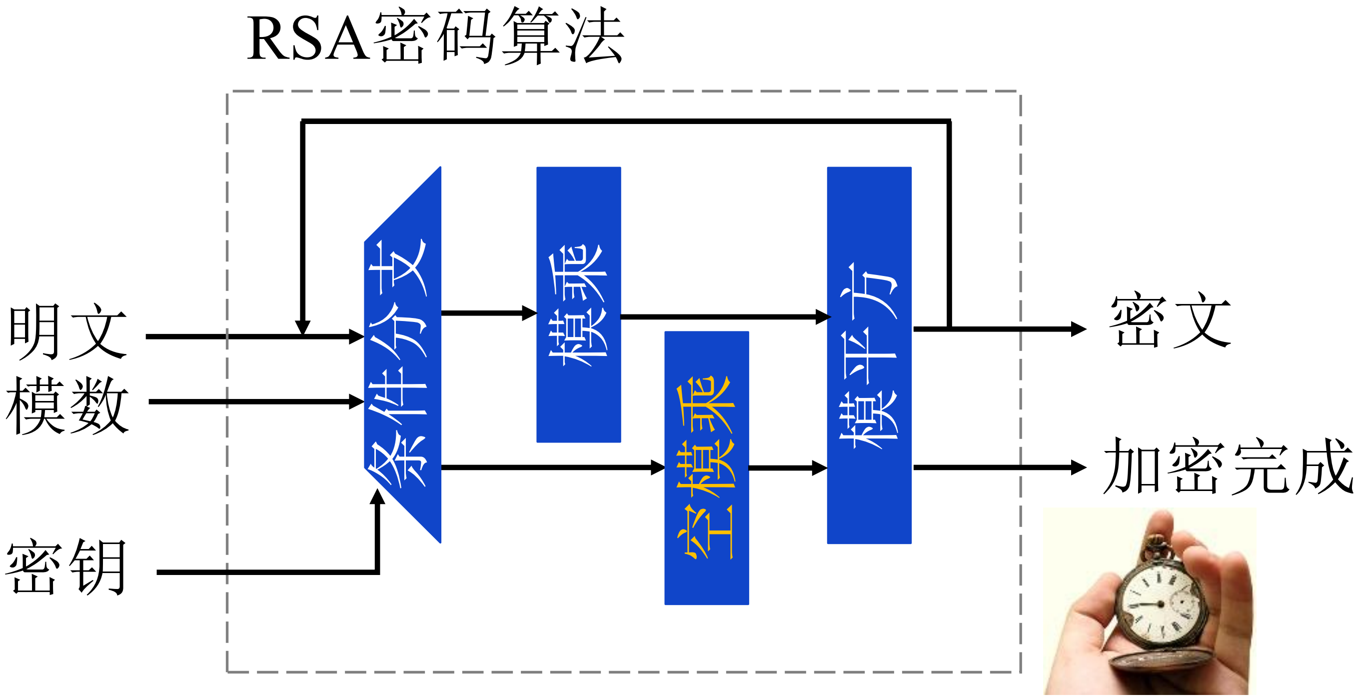
加随机延迟是否有效?

RSA密码算法



5.7(1) RSA时序防御方法

- 加空操作



5.7(1) RSA时序防御方法

Protection Techniques Against Timing Attack

- Montgomery Power Ladder： $M = C^d \bmod n$

```
r0 := 1; r1 = C
for i := w − 1 to 0 do
    if d[i] == 1 then
        r0 := r0 * r1; r1 := (r1)^2
    else
        r1 := r0 * r1; r0 := (r0)^2
    end if
end for
return r0
```

```
s[w] := 1
for i := w − 1 to 0 do
    if d[i] == 1 then
        m[i] := s[i + 1] * c mod n
    else
        m[i] := s[i + 1] * 1
    end if
    s[i] := m[i] * m[i] mod n
end for
return m[0]
```

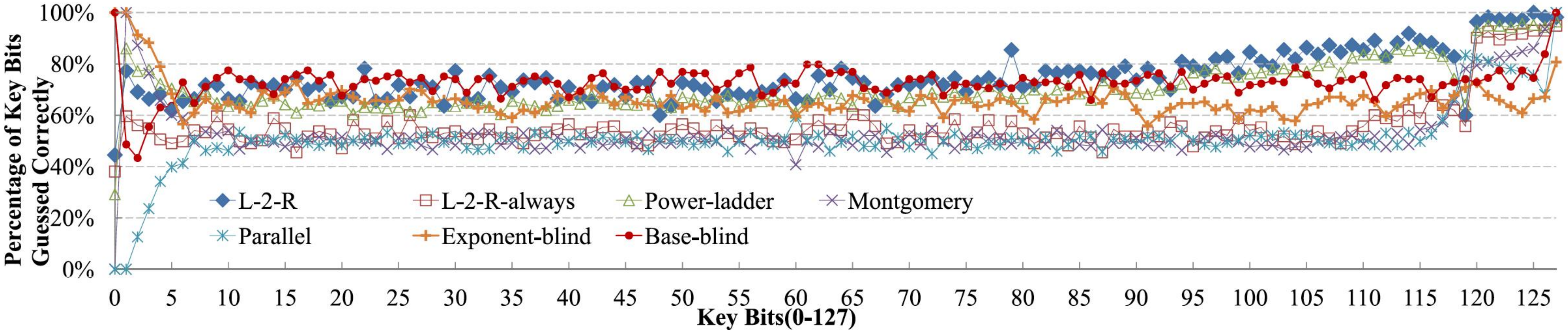
5.7(1) RSA时序防御方法

- 优化模乘$x \cdot y\ mod\ \mathrm{N}$运算中取模环节
- 原始取模操作:
  - $x \cdot y$ 每次减去N
  - 需要减N的次数与乘积结果相关（求模运算时间与输入有关）
- 蒙哥马利算法:
  - 基本思路是通过变换，将需要取模的数控制到很小的范围
  - 由 [0, N^2 - 2N + 1]变为 [0，2N-1]，即不超过2N
  - 这样只需要通过最多一次减法即可完成取模运算（求模运算时间均衡化）

```
for i := w − 1 to 0 do
    if d[i] == 1 then
        m[i] := s[i + 1] * c mod n
    else
        m[i] := s[i + 1] * 1
    end if
    s[i] := m[i] * m[i] mod n
end for
```

https://blog.csdn.net/zhushuanghe/article/details/121940152

5.7(2) 不同RSA架构攻击结果



课后阅读

- P. C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” Advances in Cryptology - CRYPTO’96, Springer-Verlag Lecture Notes in Computer Science, vol. 1109, pp. 104– 113, 1996.
- RSA Blinding, https://www.openssl.org/docs/man1.0.2/man3/RSA_blinding_on.html
- Linux man page, https://linux.die.net/man/3/rsa_blinding_on

  https://linux.die.net/man/3/rsa

感谢聆听!

THANK YOU FOR YOUR ATTENTION!