黑客攻防

公钥密码

Unit04

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	知识讲解
	10:30 ~ 11:20	
	11:30 ~ 12:00	
下午	14:00 ~ 14:50	实训案例
	15:00 ~ 15:50	
	16:00 ~ 16:50	扩展提高
	17:00 ~ 17:30	总结和答疑

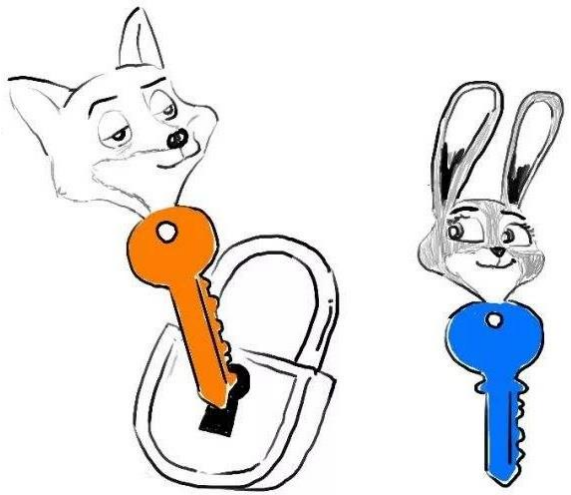
知识讲解


知识讲解
公钥密码的概念
公钥密码的特点
RSA算法的原理
公钥密码的概念
公钥密码的特点
生成密钥
公钥加密
私钥解密
运算代价

公钥密码的概念

公钥密码的概念

- 传统对称密码体制要求通信双方使用相同的密钥，因此应用系统的安全性完全依赖于密钥的保密
- 公钥密码体系，亦称非对称密码体系，加密和解密使用两把不同的密钥，加密算法和加密公钥可以公开，但是解密私钥必须保密，只有解密方知道
- 公钥密码体系要求算法必须保证，任何攻击者都无法从公钥推算出私钥
- 公钥密码体系中最著名的是RSA算法，此外还有背包密码、McEliece、Diffie-Hellman、Rabin、零知识证明、椭圆曲线以及ElGamal等多种算法

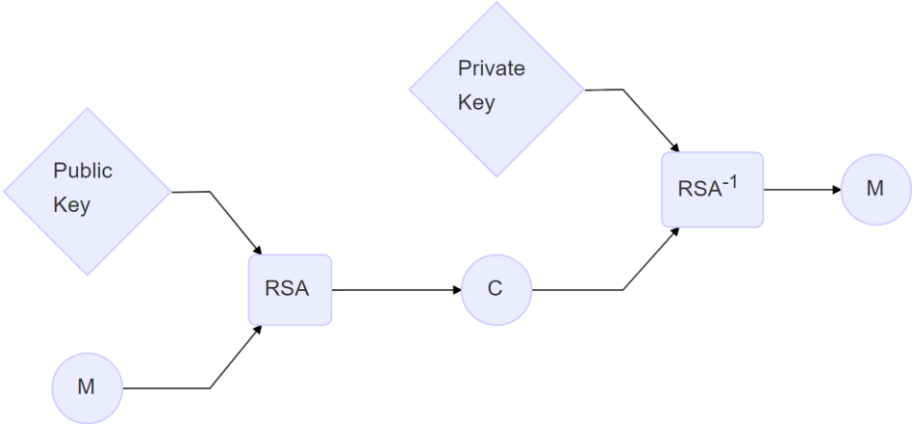


知识讲解

公钥密码的特点

公钥密码的特点

- 明文(M)：作为算法输入的消息或数据
- 加密(RSA)：对明文进行各种代换和变换
- 密文(C)：作为算法输出，看似完全随机而杂乱的数据，依赖明文和秘钥：
  - 对于给定的消息，不同的密钥产生不同的密文
  - 密文是随机的数据流，其意义是无法被理解的
- 密钥(Key)：公钥(Public Key)和私钥(Private Key)成对出现，一个用来加密，另一个用来解密
- 解密(RSA^{-1})：接收密文，解密还原出明文
- 公钥密码体系如下图所示：



知识讲解

RSA算法的原理

生成密钥

- 生成公私密钥对
  - 任选两个质数p和q，二者相乘得到n，即$n = p \times q$。如$p = 3$，$q = 11$，$n = 3 \times 11 = 33$
  - 由欧拉函数$\phi(n) = (p-1) \times (q-1)$得到在$[1, n)$区间内与$n$互质的正整数个数。如$\phi(33) = (3-1) \times (11-1) = 20$，即在[1,33)区间内有1、2、4、…、32共20个正整数与33互质
  - 在$[1, \phi(n))$区间内选择一个与$\phi(n)$互质的正整数e，得到公钥$\{e, n\}$。如在[1,20)区间内选择与20互质的正整数3，得到公钥{3,33}
  - 确定正整数d，满足$d \times e \equiv 1\%\phi(n)$，即方程$(d \times e - 1)\%\phi(n) = 0$的解，得到私钥$\{d, n\}$。如$(7 \times 3 - 1)\%20 = 0$，得到私钥{7,33}。式中符号“≡”表示同余，$a \equiv b\%c$表示$a$和$b$除以$c$所得余数相等

知识讲解

公钥加密

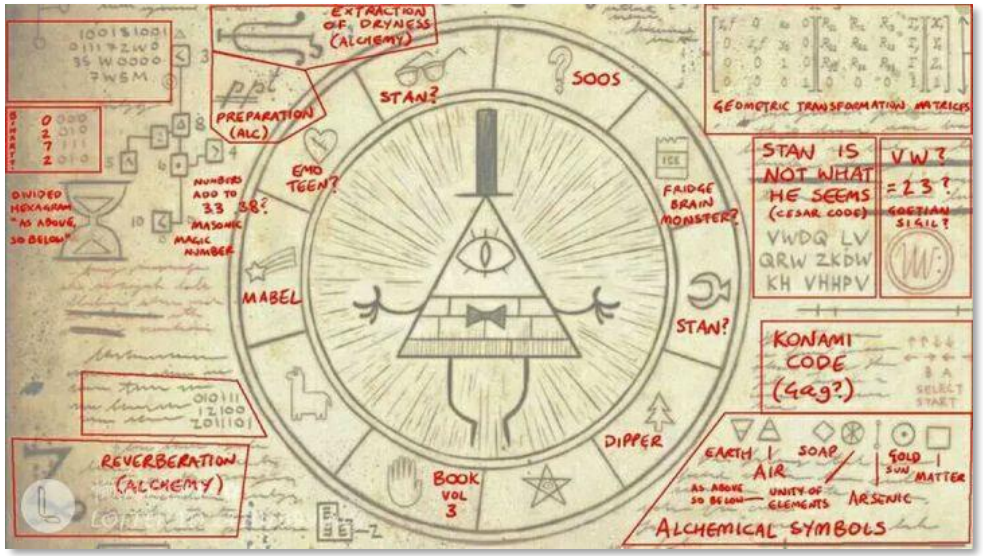
- 利用公钥$\{e,n\}$将明文分组M加密成密文分组C
  - $C = M^e \% n$

知识讲解

私钥解密

- 利用私钥$\{d,n\}$将密文分组C解密成明文分组M
  - $M = C^d \% n$

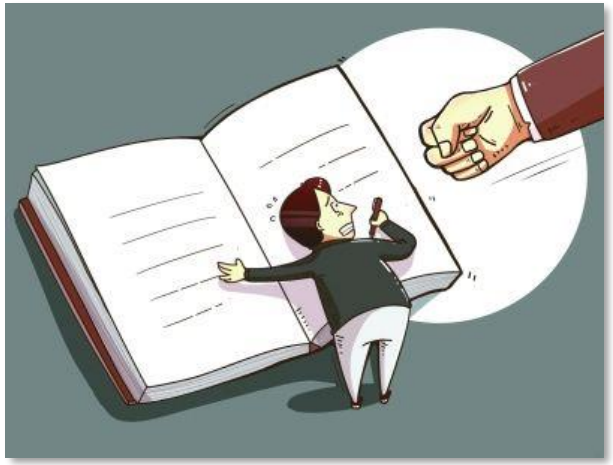
知识讲解



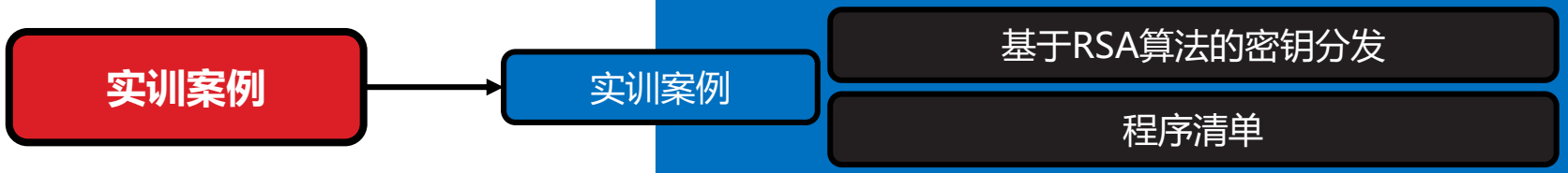
运算代价

- 运算代价高，速度比对称加密慢几个数量级，不适合加密大量数据
  - 用公钥加密私钥解密数字内容的对称密钥——防窃取、防监听
  - 用私钥加密公钥解密数字内容的消息摘要——防篡改、防抵赖

知识讲解



实训案例



实训案例

基于RSA算法的密钥分发

- 在第三单元“基于DES加密的TCP聊天”案例的基础上进行二次开发，利用RSA算法生成的非对称密钥加密和解密DES密钥
- 在Linux平台上完成基于RSA算法加密通信程序的设计和实现
- 程序包含DES密钥自动生成、RSA密钥分配和DES加密通信等三个部分
- 程序实现全双工通信，并且加密过程对用户透明
- 借助多路I/O或异步I/O技术，对网络通信功能的实现进行优化

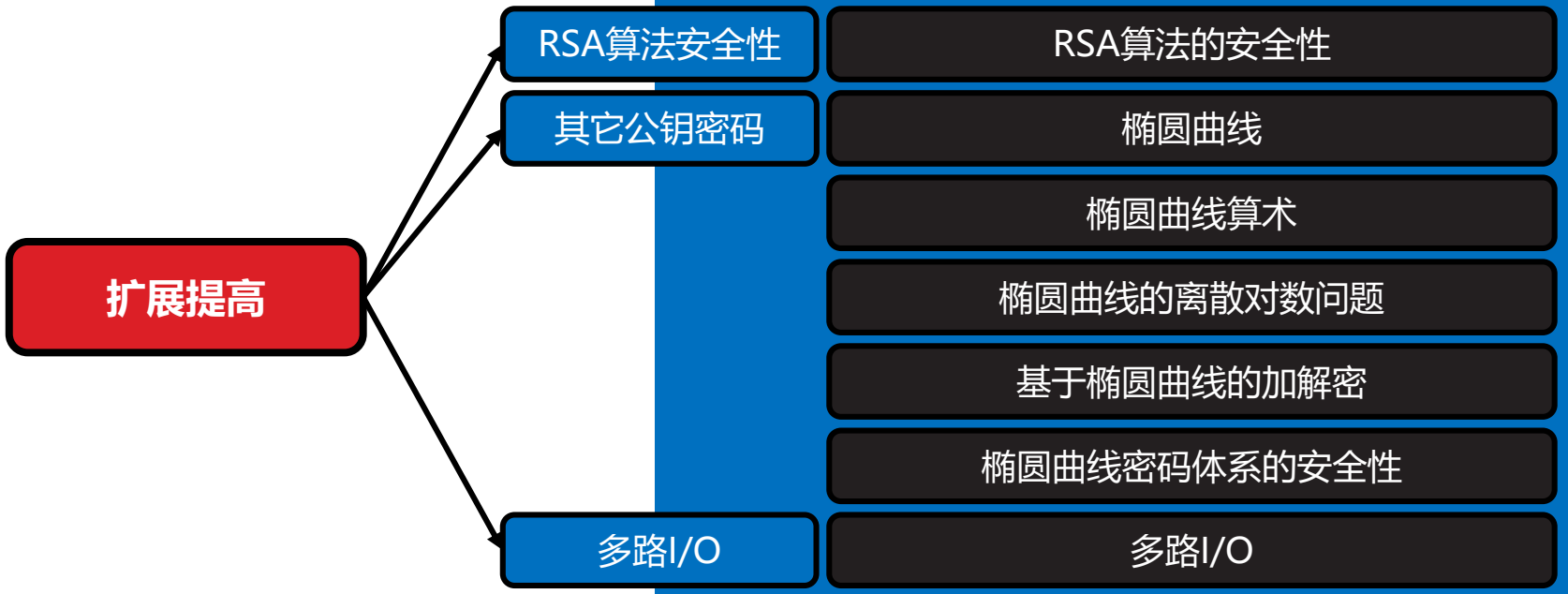
代码实践

程序清单

- 声明Rsa类
  - rsa.h
- 实现Rsa类
  - rsa.cpp
- 测试Rsa类
  - rsa_test.cpp
- 测试Rsa类构建脚本
  - rsa_test.mak
- 声明MoreSecChat类
  - moresecchat.h
- 实现MoreSecChat类
  - moresecchat.cpp
- 测试MoreSecChat类
  - moresecchat_test.cpp
- 测试MoreSecChat类构建脚本
  - moresecchat_test.mak

代码实践

扩展提高


扩展提高
RSA算法安全性
其它公钥密码
多路I/O
RSA算法的安全性
椭圆曲线
椭圆曲线算术
椭圆曲线的离散对数问题
基于椭圆曲线的加解密
椭圆曲线密码体系的安全性
多路I/O

RSA算法的安全性

RSA算法的安全性

- 目前针对RSA算法的攻击主要还是利用质因子分解法
  - RSA算法的公钥$\{e,n\}$公开，将其中的n分解为两个质数p和q的乘积，即$n=p\times q$
  - 根据n的两个质因子p和q计算其欧拉函数$\phi(n)$的值，即$\phi(n)=(p-1)\times(q-1)$
  - 由已知的e求出满足方程$(d\times e-1)\%\phi(n)=0$的d，得到私钥$\{d,n\}$，完成破解
- RSA算法的安全性依赖于对n的质因子分解
  - 当n足够大(如1024或2048个二进制位)时，迄今尚无工具可从中分解出p和q因子
- 目前从理论上还无法证明，大数分解是破解RSA算法的唯一途径
  - RSA算法的安全性是否与大数分解的难度完全等价，缺乏理论上的支持
  - 纵观近30年历史，RSA算法经受住了各种攻击的考验，其安全性已被事实所证明

知识讲解

其它公钥密码

椭圆曲线

- 椭圆曲线密码学(Elliptic Curve Cryptography，ECC)是基于椭圆曲线数学的一种公钥密码算法
- 这里的椭圆曲线并非圆锥曲线意义上的椭圆曲线：

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1 \quad (a>b>1)$$

- 而是源自用于计算椭圆周长的椭圆积分：

$$\int_c^x R\left[t,\sqrt{P(t)}\right]dt$$

- 其中R是一个有理函数，P是一个无重根的3或4阶多项式，而c是一个常数

知识讲解

椭圆曲线

知识讲解

- 椭圆曲线方程与P的表现形式比较相像。数学上的椭圆曲线是指由维尔斯特拉斯(Weierstrass)方程所确定的平面曲线，其一般形式为：

$$y^2 = x^3 + ax^2 + bx + c$$

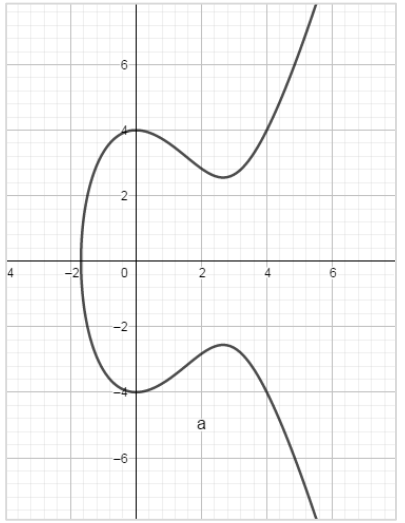
  - 其中：

$$\Delta(E) = -4a^3c + a^2b^2 - 4b^3 - 27c^2 + 18abc \neq 0$$

- 典型的椭圆曲线，如：

$$y^2 = x^3 - 4x^2 + 16$$

  - 其函数图像如下所示：



- 在密码学中使用的椭圆曲线方程一般被限定为：

$$y^2 = x^3 + ax + b$$

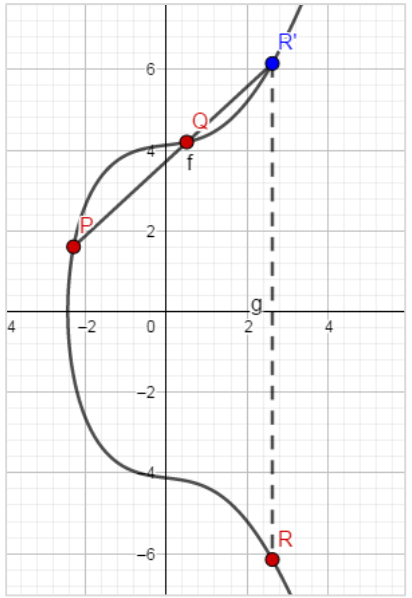
  - 即二次项系数为0，其中：

$$\Delta(E) = 4a^3 + 27b^2 \neq 0$$

椭圆曲线算术

知识讲解

- 加法
  - 已知椭圆曲线上两个不同的点P和Q，它们的和$R = P + Q$，可依如下方法得到：
    - 过P、Q两点做直线L，与椭圆曲线相交于R'点
    - R'点关于X轴的对称点即为P、Q两点之和R点
    - 椭圆曲线关于X轴对称，故R点亦在椭圆曲线上
  - 如下图所示：
  - 另外，$-P$点是P点关于X轴的对称点，亦在椭圆曲线上

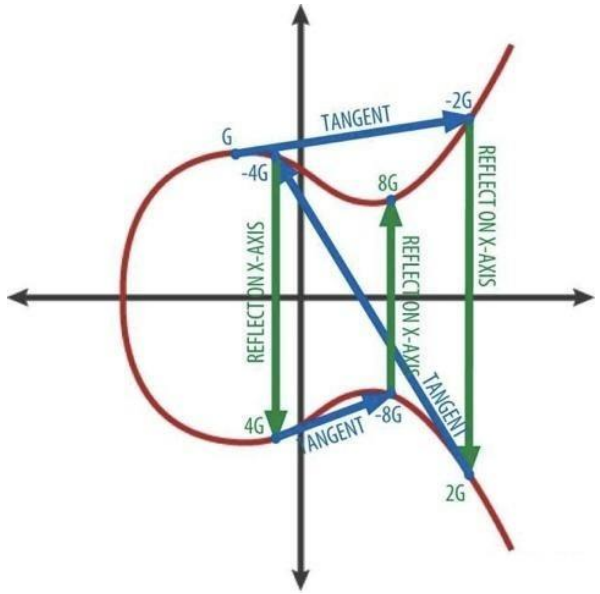


- 数乘
  - 移动Q点至与P点重合，直线L即成为椭圆曲线上过P点的切线
    - 此时的R点为P点与P点之和，即$R = P + P = 2P$
    - 依此类推可以得到$3P$、$4P$、……，直至kP $(k \geq 1)$
    - 此即椭圆曲线的数乘

椭圆曲线的离散对数问题

- 假设数k和P点已知，可以很容易地计算其数乘的结果$Q = kP$，但反过来在已知P点和Q点的前提下，计算数k却相当困难，此即椭圆曲线的离散对数问题
- 正因为如此，可以将Q作为公钥，公开出去，k作为私钥，秘密保管，通过公钥破解私钥将十分困难

知识讲解



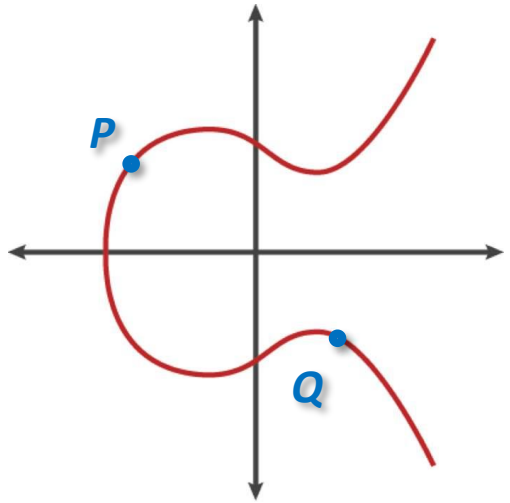
基于椭圆曲线的加解密

- 应用于密码体系中的椭圆曲线都是定义在有限域上的，即：

$$y^2 \equiv (x^3 + ax + b)\%p$$

  - 其中p称为模数

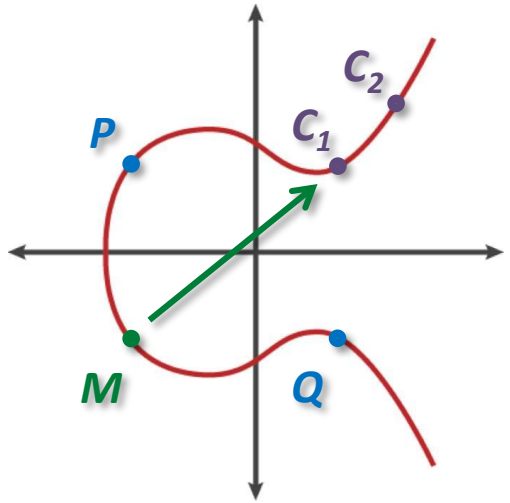
- 生成公私密钥对
  - 在椭圆曲线$y^2 \equiv (x^3 - 4)\%199$上选取$P = (2,2)$和数$k = 119$，计算$Q = kP = 119(2,2) = (183,173)$，得到：
    - 公钥：$Q = (183,173)$
    - 私钥：$k = 119$



知识讲解

基于椭圆曲线的加解密

- 加密运算
  - 将明文76代入椭圆曲线方程$y^2 \equiv (x^3 - 4)\%119$，得到明文点：
    - $M = (76,66)$
  - 随机选取一个正整数$n = 133$，利用公钥依下式根据明文点计算密文点：
    - C
      $= \{C_1, C_2\}$
      $= \{nP, M + nQ\}$
      $= \{133(2,2), (76,66) + 133(183,173)\}$
      $= \{(40,147), (180,163)\}$

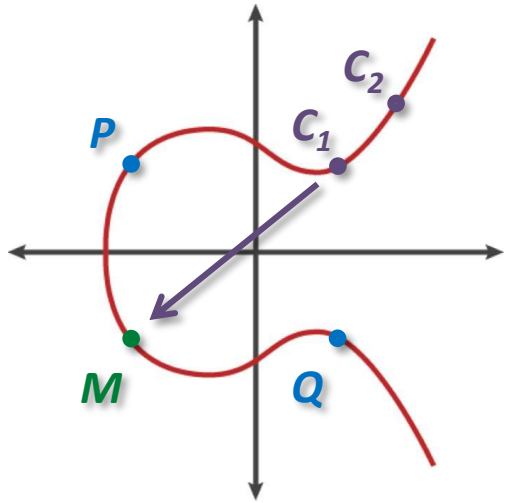


知识讲解

基于椭圆曲线的加解密

- 解密运算
  - 利用私钥依下式根据密文点计算明文点：
    - M
      $= C_2 - kC_1$
      $= (180,163) - 119(40,147)$
      $= (180,163) - (98,52)$
      $= (76,66)$
  - 由明文点$(76,66)$得到明文76

知识讲解



椭圆曲线密码体系的安全性

- 椭圆曲线密码体系是截至目前每比特加密强度最高的一种公钥密码体系。破解椭圆曲线密钥的时间复杂度是指数级($O(2^N)$)的，远高于子指数级时间复杂度的RSA算法：
  - 有研究表明160位椭圆曲线密钥的安全强度与1024位RSA密钥相当
  - 在一百万条指令每秒的处理器上，破解234位椭圆曲线密钥，需要$1.6x10^{28}$年
- 我国国家标准对各种加密算法的最小密钥长度做了如下规定：
  - 对称分组加密：80位
  - RSA加密：1024位
  - 椭圆曲线加密：160位

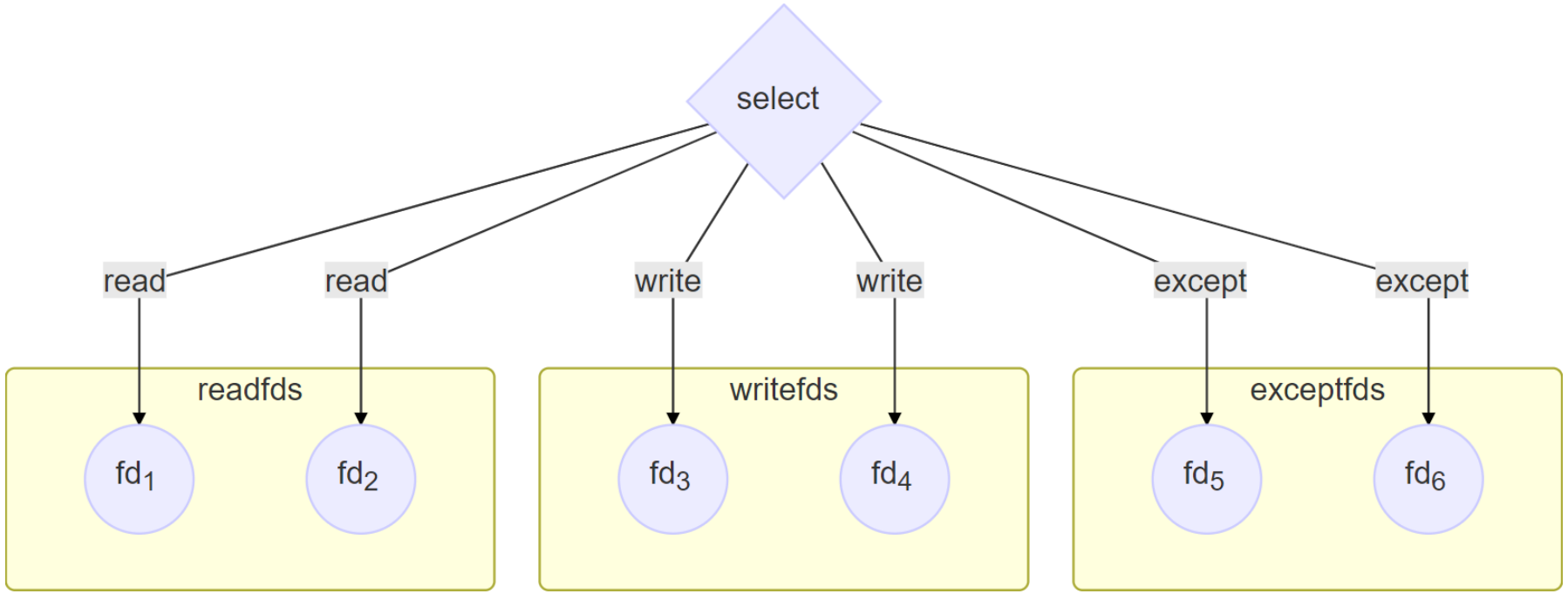
知识讲解

多路I/O

多路I/O

- 为了提高程序的运行效率，Linux系统提供了select函数。借助该函数，调用者可在一个线程中同时监视多个文件描述符上的I/O事件，哪个文件描述符读/写就绪了就读/写哪个文件描述符，而不必为每个可能发生阻塞的文件描述符开辟单独的进程或者线程。如下图所示：

知识讲解



多路I/O

- select函数的原型如下所示：

```
#include <sys/select.h>

int select(
    int             nfds,      // 被监视文件描述符中的最大值加一
    fd_set        * readfds,   // 监视这些文件描述符上的可读事件
    fd_set        * writefds,  // 监视这些文件描述符上的可写事件
    fd_set        * exceptfds, // 监视这些文件描述符上的异常事件
    struct timeval * timeout    // 在此时间内未发生任何事件则返回
);
```

- 该函数成功返回有事件发生的文件描述符总数，或者0如果超时，失败返回-1，同时设置errno变量

知识讲解

多路I/O

- 用于操作文件描述符集的四个便捷宏：

```
void FD_CLR   (int fd, fd_set * set); // 清除文件描述符集中的特定文件描述符
int  FD_ISSET (int fd, fd_set * set); // 特定文件描述符是否在文件描述符集中
void FD_SET   (int fd, fd_set * set); // 将特定文件描述符加入文件描述符集中
void FD_ZERO  (        fd_set * set); // 清除文件描述符集中的全部文件描述符
```

- 表示超时时间的结构：

```
#include <sys/time.h>

struct timeval {
    long tv_sec;  // 秒
    long tv_usec; // 微秒(1/1000000秒)
};
```

- 调用select函数时将timeout参数指定为NULL，表示无限期等待事件发生

知识讲解

总结和答疑