第三次作业

（1）测试点 3-1

口令检测程序，要求：必须包含大写字母，小写字母，数字，特殊字符中的三种，要求长度 8 到 30 位。

语言：js，思路：使用正则表达式验证的方法，

核心代码

```
watch: {
  // eslint-disable-next-line vue/no-dupe-keys
  password: function () {
    // eslint-disable-next-line max-len
    if (/^(?![a-zA-Z]+$)(?![A-Z0-9]+$)(?![A-Z\W_!@#$%^&*`~()-+=]+$)(?![a-z0-9]+$)(?![a-z\W_!@#$%^&*`~()-+=]+$)(?![0-9\W_!@#$%^&*`~()-+=]+$)[a-zA-Z0-9\W_!@#$%^&*`~()-+=]{8,30}$/.test(this.password)) {
      this.tip = '口令符合格式'
    } else {
      this.tip = '口令不符合格式，数字大小写字母特殊字符四者之三，8 到 30 位'
    }
  }
}
```

结果：

（2）测试点 3-2

1. needhamschroeder 协议缺陷和改进

中间人攻击：中间人在第三步 A->B 发送消息时，截获消息，并伪装 B 向 A 发送一个格式相同的随机数，A 使用 Kab 解密，等同于加密，然后发送给 B，中间者随即也能截获。

重放攻击：攻击者可以直接跳到第三步发送一个旧的 Kab 给 B，从而不需要认证就能和 B 进行通信。

改进方案：造成缺陷的原因是不确定消息是否新鲜，以及 B 无法认证 A,所以需要加入时间戳以及公钥证书。

2.公钥密码的 Needham-Schroeder 的分析。

风险：攻击者可以两次运行 NS 公钥协议进行攻击。

攻击者第 1 次 NS 公钥协议运行

消息 1.1：A→Z(A)：{Na，A}Kz

此时，入侵者 Z 开始第 2 次 NS 协议运行：

消息 2.1：Z(A)→B：{Na，A}Kb

消息 2.2：B→Z(A)：{Na，Nb}Ka

消息 1.2：Z(A) →A：{Na，Nb}ka

消息 1.3：A→Z(A)：{Nb}Kz

消息 2.3：Z(A)→B：{Nb}Kb

入侵者 Z 通过解密消息 1.1 和消息 1.3 获取发送消息 2.1 和消息 2.3 所需的 Na 和 Nb，消息 1.2 则是消息 2.2 的重放上述协议运行完，主体 B 认为他与 A 共享秘密 Nb，实际上他与共享 Nb，Z 假冒 A 成功，攻击有效.对于网络系统中任何一个合法用户，只要接收到发给自己的 NS 公钥协议消息 1，就可以发起上述攻击，以欺骗另外一个用户，故 NS 公钥协议是不安全的。

改进方案：

A→B：{Na，A}Kb

B→A：{Na，Nb，B}Ka

A→B：{Nb}Kb