CSCI 585: Information Security and Privacy

**🔹 1. 信息安全基础**

* 了解 CIA 模型（三要素）

1. 应用的本质是保护数据 ，安全的本质是数据安全的使用
2. Integrity, confidentiality, Availability
   1. Confidentiality机密性，数据不可见，确保数据制备授权的主体访问
   2. Integrity完整性，数据不可修改，确保授权修改
   3. Availability可用性，数据可读，确保授权可读

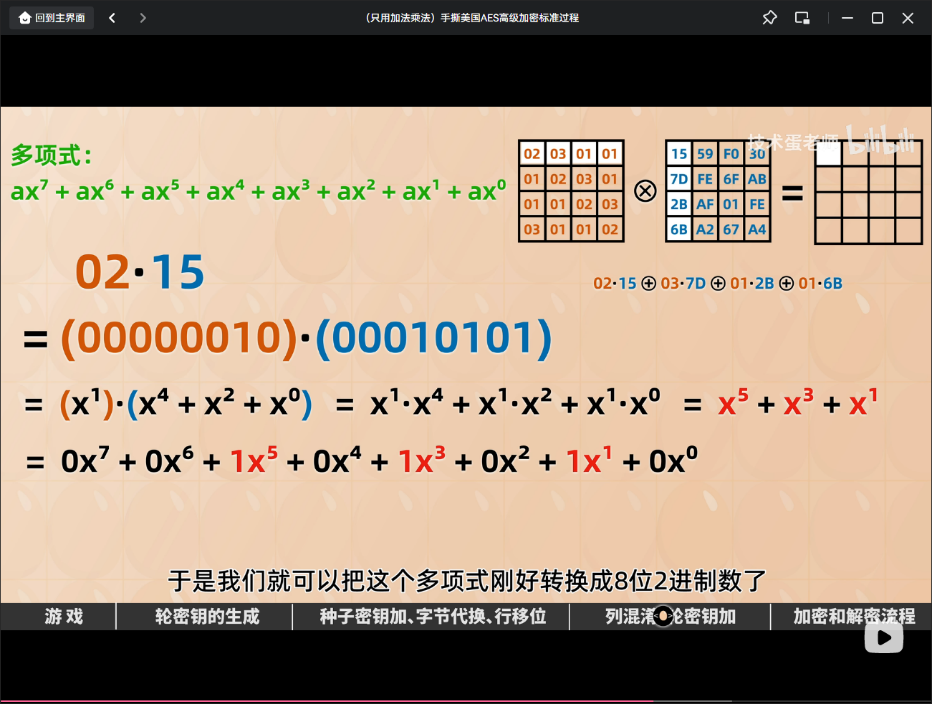
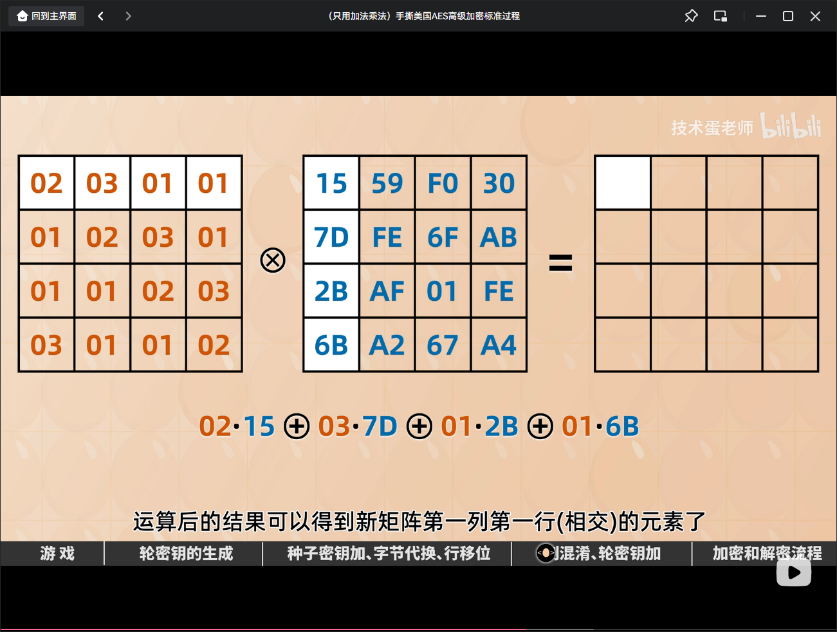
* 简单了解 Hash、AES、RSA 是做什么的（不用太深入）
  1. Hash 把无限多的字符转化问有限的数字，不可逆。
     + 冲突解决：
       - 封闭寻址法，把冲突数据放到该数据的链表里面，可能会发生集群现象（数据会聚集在某个节点里面）图片包含 图表

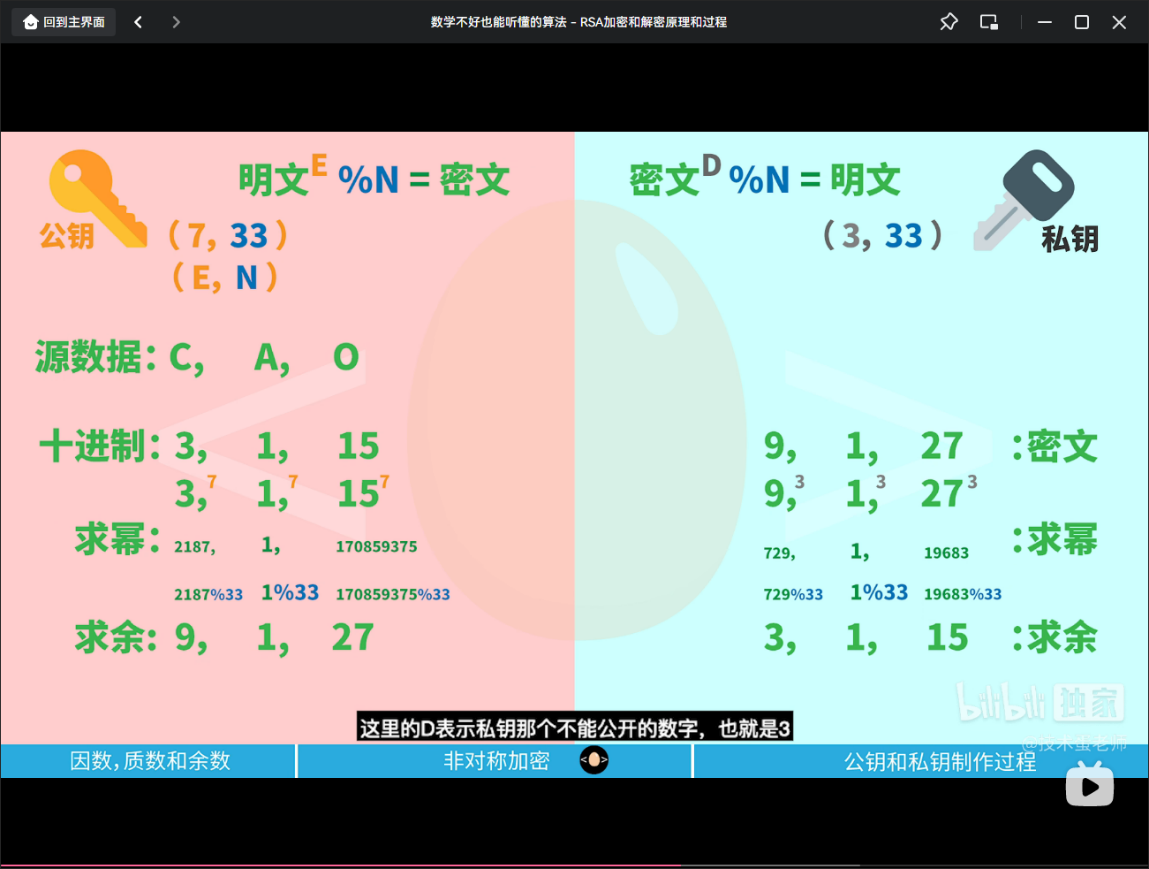
         AI 生成的内容可能不正确。
       - 开放寻址法，发生冲突就往下面找空节点，2次找坑画着卡通人物

         AI 生成的内容可能不正确。还有等等方法、
     + 哈希算法（运算为固定的值：摘要）：
       - MD5: 128bit
       - SHA-256:256bit
       - SHA-512:512bit
     + 为什么要用hash：如果数据库被黑，黑客也看不见数据
  2. AES
     + 在加密前需要先把数据进行分组，128bit每组（不足填充），这128bit数据用4×4的矩阵排列，从上到下从左到右顺序电脑萤幕

       AI 生成的内容可能不正确。Aes加密还需要把字符转化成16进制，两个16进制数可以占一个格子
     + 轮密钥：4的被数列旋转，再查看s-box中所对应的行和列替换数字，公式：(列\_i-4) ⊕字节替换(s-box)（旋转字节\_(i-1)）⊕（轮常量[每轮一个固定常数（如 0x01, 0x02, 0x04, ...）]），非4的倍数（列\_i-4）⊕(列\_i-1)比如：5的倍数就是4列⊕1列
     + 种子密钥加：明文和种子密钥进行⊕，
     + 加密操作
       - 字节代换：然后再用s-box替换，
       - 再行移位：每一行向左移动自己的行数，每行移动完后把超出矩阵原本范围的格子往回挪动表格

         AI 生成的内容可能不正确。文本, 表格

         AI 生成的内容可能不正确。，
       - 之后列混淆，把得到的行移位和固定矩阵进行模×运算（「当前整列4个字节 × 一个固定矩阵的一行」）遇到x^8:卡通人物

         AI 生成的内容可能不正确。,
       - 轮密钥加：用列混淆和第一轮的轮密钥进行异或运算
       - 第2-9轮的加密：把上一轮得出的矩阵进行字节代换，再行移位，之后列混淆，轮密钥加…,第10轮不用列混淆
     + 解密：逆向操作，用第十轮的轮密钥
  3. RSA
     + 因数，质数，余数
       - 因数：两个相乘的数a×b=c
       - 质数：a=1×a只能由数字1和本身相乘得出来的数
       - 余数：a%b=c
     + 非对称加密
       - 公钥加密数据发送数据，接收方是所有人都知道，私钥是只有接收方有，比如公钥（7,33）私钥(?,33),
     + 公钥私钥制作过程
       - 2个质数 p=3;q=11
       - 质数相乘 N=p\*q = 3\*11 = 33
       - 欧拉函数 T=(p-1)\*(q-1) = 2\*10 = 20
       - 选公钥E：质数;1<公钥<T；不是T的因子 E = (3,33)
       - 算私钥D : (D\*E)%T=1 D=(7,33)
* 学习什么是访问控制、用户认证（密码 vs 双因子）
  1. 访问控制
     + 概述
       - 概念：对资源对象访问者授权，控制方法以及运行机制，访问者是主题；授权（方式）是访问者可以对资源对象进行访问的方式；控制（能不能干）是对访问者使用方式的检测和限制：
       - 访问控制目标：1.防止非法用户进入系统，2.阻止合法用户对系统资源的非法使用。
       - 要实现访问控制的目标：1.对网络用户进行有效身份认证2.进行系统的安全审计和监控，检测用户对系统的攻击企图
     + 模型
       - 由一组安全机制组成：主体（Subject），参考监视器（Reference Monitor）（访问控制的决策单元和执行单元的集合体，负责做出访问决策并执行控制，还负责监控并记录事件），客体（Object），访问控制数据库（权限规定（ACDB）：记录主体访问客体的权限以及其他方式的信息），审计库（访问行为记录（Audit Log）：存储主体访问客体的操作信息）
     + 访问控制类型：
       - 自主访问控制DAC：
         * 客体的所有者按照自己的安全策略授予系统中的其他用户对其的访问权
         * 主体附表、能力表、ACL（访问控制列表）等方式是常见实现形式。
       - 强制访问控制（MAC）：
         * 系统根据主体和客体的安全级别（如机密等级）强制执行规则，用户不能绕过
       - 基于角色访问控制（RBAC）：
         * 用户被赋予角色，角色决定权限，无需为每个用户单独配置权限
       - 基于属性的访问控制（ABAC）:
         * 通过检查主体、客体及环境的属性（如时间、位置等）动态判断授权
       - 其他模型:
         * 包括基于行为、时间、关系（ReBAC）等更细粒度的访问控制形式
     + 实施流程
       - 身份认证（Authentication）：确认访问者是谁
         * 方式包括：密码、生物识别、卡片等
       - 权限判定（Authorization）：该用户有没有权限访问资源
         * 参考模型如 RBAC、MAC 等
       - 执行控制（Enforcement）：参考监视器执行允许或拒绝操作
       - 行为审计（Auditing）：记录访问过程，便于事后分析和问责
     + 安全设计原则
       - 最小权限原则（Least Privilege）：只给完成任务所需最少权限
       - 默认拒绝原则（Default Deny）：默认不允许访问，需明确授权
       - 分离职责（Separation of Duty）：防止单点控制
       - 御深度（Defense in Depth）：多层安全防护机制
     + 应用场景举例
       - 操作系统中：
         * Linux文件权限（rwx）、SELinux使用MAC模型
       - 数据库中：
         * 用户/表/字段级别的权限控制，如 GRANT、REVOKE
       - Web系统中：
         * 登录认证（Auth）、基于角色/属性进行授权

**🔹 2. 常见攻击原理**

* 了解 SQL 注入、XSS 是什么
  + SQL注入
    - 原理： 故意写错填空内容，使得服务器向数据库的正常查询变成了不正常的代码执行
    - 报错与注释
      * 日程表

        AI 生成的内容可能不正确。
      * Eggpain’ –
      * 假OR真（1=1）--
    - UNION
      * 图片包含 图形用户界面

        AI 生成的内容可能不正确。
      * 图形用户界面

        AI 生成的内容可能不正确。
    - 布尔和时间延迟
      * Cookie文本

        AI 生成的内容可能不正确。图形用户界面, 网站

        AI 生成的内容可能不正确。图片包含 文本

        AI 生成的内容可能不正确。
      * 延迟 sleep:正常响应就是猜中，超过10秒就寄
    - LOAD\_FILE():提供域名查询DNS，黑客把要查询的管理员密码和域名进行拼接
  + XSS攻击
    - 反射型
      * 图形用户界面

        AI 生成的内容可能不正确。
    - 储存型
      * 图示, 示意图

        AI 生成的内容可能不正确。
    - DOM型
      * 图形用户界面

        AI 生成的内容可能不正确。
  + IDOR漏洞的原理，实战和预防
    - 越权
      * 越权访问不属于自己的资源出现IDOR漏洞图片包含 图示

        AI 生成的内容可能不正确。
    - IDOR
      * Direct Object References直接对象引用
      * 比如在服务器上传一份文件，这个文件在服务器那里被称为一个对象，应用以某种方式暴露这份文件比如url，我们访问url就能看到这个文件
      * I是insecure
    - 防范：
      * 哈希处理：把暴露出来的数字没有那么容易预测
  + CSRF攻击
    - 无状态和cookie
      * http协议无状态，不保存上一次请求
      * 服务器在收到账号以后核对没问题就会生成一串字符，放在set-cookie，浏览器以后发送请求时只需要把这串字符放在cookie里面供服务器验证
    - CSRF（cross site Request Forgery）发生过程
      * 通过f12调试功能复制form功能，复制粘贴到黑客电脑上文本

        AI 生成的内容可能不正确。
        + 先把action给补全（原网站的url）
        + 修改value
        + 最后交脚本自动提交屏幕上有字

          AI 生成的内容可能不正确。
      * 黑客发送给你图示

        AI 生成的内容可能不正确。
    - 如何防御
      * 使用新的浏览器
      * CSRF token
        + 加入imput标签，每个用户有不同的token值，会加入http的身体里
      * Samesite cookie
        + Strict:给浏览器加入新的属性，不允许跨站传输cookie
        + Lax:浏览器允许跨站发送cookie

允许顶级导航 = “直接把浏览器主窗口跳转到另一个网站”

“脚本偷偷发请求、iframe嵌套” ≠ 顶级导航

* 看几个基础网络攻击演示视频（推荐搜索“OWASP Top 10”）

**🔹 3. 阅读经典安全入门资料**

* 推荐入门书籍：《Computer Security: Principles and Practice》by Stallings（看看前两章）
* 推荐网站：[OWASP](https://owasp.org/): 免费资源讲各种安全问题

**🔹 4. 编程与实践（选做）**

* 简单 Python 编程（如用 socket 写个客户端、模拟登录流程）
* 安装 VirtualBox + Kali Linux，熟悉基本安全实验环境