1. 数据加密

- 加密原理介绍

- 数据存储加密

- 数据库存储加密：通过密文存储数据，并配合存取控制机制，有效防范数据在存储环节的泄露风险，确保数据库中数据的机密性和完整性。
- HTML 存储加密：作为网页防篡改技术，在众多领域如电子商务、电子政务等保障信息安全，防止网页内容被非法篡改。
- 磁盘空间加密：对整个磁盘或特定分区进行加密操作，是保护系统安全的重要手段，适用于各类操作系统，可防止磁盘数据被未授权访问。

- 数据传输加密

- 传输信源加密：运用密码技术对传输的信息进行加密处理，确保信息在传输过程中的保密性，防止信息被窃取或篡改。
- P2P 与 TLS：P2P 协议采用端到端加密方式，结合 TLS 协议，为网络通信提供了更高层次的安全保障，广泛应用于文件共享、即时通讯等领域。
- Telnet 与 SSH：Telnet 协议在传输过程中以明文形式传输数据，存在较大安全风险；SSH 协议则通过加密技术，可用在远程登录和数据传输。

- 数据存储加密实践

- 数据内容加密存储：利用 MySQL 内置函数，实现对数据的加密存储和解密读取操作，保障数据在数据库中的安全性。
- 磁盘空间加密：借 BitLocker 软件，对磁盘空间进行加密设置，防止磁盘数据在物理层面被非法访问。

- 数据传输加密实践

- P2P 协议传输：

（1）基本原理与特点

- P2P（Peer-to-Peer）协议是一种分布式网络传输协议，网络中的各个节点（对等方）既是客户端又是服务器，它们之间可以直接进行数据交互和共享，无需依赖中央服务器。
- 去中心化(使得网络具有更好的鲁棒性,即使部分节点失效,整个网络仍能正常工作)、自组织性、高扩展性、资源利用率高。

（2）加密传输形式与安全性

- P2P 协议支持加密传输，通过端到端的加密方式，确保数据在传输过程中的保密性和完整性。在加密传输时，数据会被加密算法（如 AES 等）处理成密文后在节点间传输，只有拥有相应解密密钥的接收方才能还原数据。(加密的 P2P 通信中，没有解密密钥，第三方无法获取数据的真实内容。同时，加密还能防止数据被篡改，保证数据的真实性。)

（3）应用场景与优势

- P2P 协议应用于文件共享（如迅雷、电驴等软件利用 P2P 技术实现快速文件下载)、即时通讯（如 Skype 早期采用 P2P 架构实现语音和视频通话)、流媒体传输（如一些网络直播

平台利用 P2P 技术减轻服务器带宽压力）等。

- Telnet 和 SSH 协议传输对比：

（1）**安全核心差异**

- **Telnet**：明文传输。
- **SSH**：加密传输，确保了数据的保密性、完整性。

（2）认证本质区别

- **Telnet**：依赖简单用户名和密码的认证方式。
- **SSH**：提供多种先进认证手段，公钥认证。公钥在服务器，私钥在用户本地，二者协同认证，即使密码泄露，无对应私钥也无法登录，大大增强了服务器的安全性防线。

- P2P 协议加密效果验证：在实现 P2P 协议加密传输后，进行抓包分析，验证加密后传输消息的保密性，即抓包无法看到传输消息，只有服务器使用私钥才能解密显示。

- Telnet 协议与 SSH 协议安全性验证：通过实验操作，明确 Telnet 协议明文传输的特点，以及 SSH 协议密文传输在保障数据安全方面的优势。

2. 数据脱敏

- 脱敏原理介绍

- 静态脱敏原理

- 替换：采用多种方式用伪装数据替换敏感数据，如使用预设的随机值、常量或根据特定规则生成的值来替代敏感信息，保护数据隐私。
- 仿真：通过算法生成新的数据，使其在保持与原始数据相似业务关联关系的同时，实现敏感数据的脱敏，确保脱敏后的数据仍能用于测试、分析等目的。
- 加密：运用密码学算法对敏感数据进行加密处理，使敏感数据以密文形式存储或传输，只有授权用户才能使用相应密钥进行解密查看。
- 遮掩：利用特定符号掩饰敏感数据的部分关键内容，如使用星号（*）代替身份证号、银行卡号等敏感信息中的部分数字，既能保护隐私，又能保留一定的数据格式特征。
- 混淆：通过打乱重排敏感数据的方式，破坏数据之间的原有关联关系，增加数据被还原的难度，有效防止敏感信息泄露。
- 偏移：针对数值型数据，采用随机移位的方法实现脱敏，使原始数据的值发生变化，但仍能保留数据的大致范围和分布特征。
- 均值化：计算敏感数据的平均值，并根据平均值对数据进行分布处理，实现脱敏效果，常用于统计分析等场景，在一定程度上保护数据隐私的同时保留数据的统计特征。
- 重采样：重新采集数据样本以代替原始数据，确保脱敏后的数据具有一定的随机性和代表性，适用于需要大量样本数据进行测试和分析的场景。
- 取整：对数值或日期数据按照指定精度进行取整操作，简化数据表示形式，同时实现一定程度的脱敏效果，常用于数据展示等场景。

- 动态脱敏原理

- 结果集解析：根据数据库表结构对查询结果进行解析判断，然后改写结果数据，这种方式

兼容性较好，但在性能方面可能存在一定瓶颈，适用于对兼容性要求较高、数据量较小的场景。
- SQL 语句改写：直接改写查询语句，使查询返回脱敏后的结果集，这种方法性能较好，但与具体的 SQL 语句耦合性较强，修改时可能需要对相关 SQL 进行较多调整。
- 混合模式脱敏技术：结合结果集解析和 SQL 语句改写两种技术的优点，平衡性能与灵活性，根据实际需求在不同场景下选择合适的脱敏方式。

- 数据静态脱敏实践

（1）如何选择合适的脱敏算法
-对于高保密要求且计算资源充足场景优先选加密算法；对于大规模数据且关系保留需求适度时，考虑高效的替换或混淆算法。

（2）如何确保脱敏后的数据可用性
- 特征保全：精心设计脱敏算法，在去除敏感信息同时最大程度保留数据统计特征（均值、方差等）和数据关联关系，保障数据在分析和业务流程中的有效性。
（3）替换算法中不同替换策略的实现原理
- 伪装数据生成：基于随机化或预设规则生成替代数据，如随机字符串、常量或根据数据分布特征生成的值，确保敏感信息被有效隐藏，同时保持数据格式与业务逻辑的一致性。
- 针对性替换：根据数据类型和业务规则，对不同敏感数据字段采用不同替换方式，如身份证号部分替换、姓名用通用名称替代等，实现精准脱敏且最小化对数据可用性的影响。

- 数据动态脱敏实践

-结果集解析

对不同数据类型适配多种脱敏算法，数值型数据可采用偏移、均值化，文本型数据可进行掩码、替换或加密处理，确保脱敏后数据保留原始特征与分布，满足数据分析需求。

-SQL 语句改写的具体实现原理

运用视图、子查询等技术构建虚拟脱敏层，无缝替换敏感数据，同时优化查询性能，确保脱敏过程高效透明，对应用程序无感知影响

-混合模式如何根据实际情况动态选择合适的脱敏策略，提高数据脱敏的效率和准确性。

- **基于多因素的智能决策**：
  - 混合模式依据数据敏感度、查询频率、性能指标等多维度因素，动态抉择脱敏策略。高频低敏查询采用结果集解析快速处理，低频高敏查询运用 SQL 语句改写深度优化。结合数据分类分级结果，对不同级别数据匹配最优算法组合，如对顶级机密数据多重加密与混淆，对普通敏感数据灵活替换与掩码，实现效率与安全的精准平衡。
- **自适应的策略调整机制**：
  - 构建实时监控体系，持续跟踪脱敏性能与数据使用模式变化。根据系统负载、用户行为变化，自动切换脱敏方式。如在业务高峰期优先保障性能，采用轻量级算法；在安全风险升高时强化安全，启用高强度加密，确保数据脱敏始终贴合实际业务需求，高效精准运行。

3. 数据访问控制

- Apache Ranger 功能
  - 集中化安全管理：提供一个集中的平台，对多个数据源和应用程序的安全策略进行统一管理，简化了安全管理的复杂性，提高了管理效率。
  - 细粒度控制：能够精确到对数据的行级、列级等进行访问权限控制，确保用户只能访问其被授权的数据，有效防止数据泄露和越权访问。
  - 统一授权方式：为不同类型的用户和应用程序提供统一的授权接口和方式，使授权管理更加规范和便捷，降低了管理成本。
  - 多种访问控制手段：支持基于角色、基于属性等多种访问控制方式，根据实际需求灵活配置访问策略，适应不同的业务场景和安全要求。
  - 集中审计：对所有的数据访问操作进行集中记录和审计，方便事后追踪和分析，有助于发现潜在的安全问题和违规行为。

4. 数据水印

- 空域水印技术 - LSB 算法原理及图像水印算法实现

- 数据水印原理

- 弥补密码学不足：数据水印技术作为密码学的补充手段，将特定的水印信息嵌入到数字化媒体（如图像、音频、视频等）中，用于标识数据的版权所有者、来源等信息，即使数据被非法获取，也能通过水印提取来证明版权归属。
- 通用方案：包括水印生成算法，根据需要生成具有特定特征的水印信息；水印嵌入算法，将水印信息巧妙地嵌入到载体数据中，使其在不影响载体数据正常使用的前提下，携带水印信息；水印提取检测算法，用于在需要时准确提取出水印信息，并检测水印的完整性和真实性。
- 密钥加强安全性：在水印生成、嵌入和提取过程中使用密钥，只有授权用户拥有正确的密钥才能进行相应操作，有效防止水印被非法篡改或伪造，提高了水印技术的安全性。
- 检测手段及错误类型：通过相关检测算法对提取的水印进行验证，判断水印是否完整、是否被篡改等。常见的错误类型包括误报（将正常数据误判为含有水印）和漏报（含有水印的数据未能检测到水印），通过优化算法和参数设置来降低错误率。
- 提高鲁棒性的方法：采用多种技术手段提高水印在数据受到各种攻击（如压缩、裁剪、滤波等）后的鲁棒性，确保水印信息仍能被准确提取，如采用纠错编码技术、选择合适的嵌入位置和强度等。

- 流载体的 LSB 算法

- 嵌入过程：在图像等流载体数据中，选择合适的载体子集（如像素点的最低比特位），将水印信息替换到这些最低比特位上，实现水印的嵌入。这种方式对载体数据的视觉影响较小，

但鲁棒性相对较弱。
- 提取过程：通过特定算法找到嵌入水印的载体子集，抽出其中的最低比特位，还原出水印信息。在提取过程中需要考虑噪声干扰、数据损坏等因素，确保水印提取的准确性。
- 解决安全问题的方法：针对 LSB 算法{通过修改载体数据（如图像、音频等）的最低有效位来嵌入水印信息。最低有效位对载体数据的感知影响较小，人眼或人耳难以察觉。}安全性较低的问题，采用密钥加密水印信息、对载体数据进行预处理、结合其他算法等方式，提高水印的安全性，防止水印被轻易去除或篡改。

- 相关算法原理

- 空域算法：如 LSB 算法及其改进算法，通过直接修改载体数据的空间域信息来嵌入水印，具有算法简单、计算量小等优点，但鲁棒性相对较弱；Patchwork 算法及其改进算法通过在载体数据中选择特定区域进行微小修改来嵌入水印，提高了水印的鲁棒性。
- 变换域算法：如 DCT 域内扩展频谱方法，将载体数据转换到变换域（如离散余弦变换域），然后在变换域中对特定系数进行修改来嵌入水印，这种方式在保证水印不可见性的同时，提高了水印对常见攻击的鲁棒性，广泛应用于图像、音频等数字媒体的水印处理。