1. 对安全多方计算了解多少?

   答：多方安全计算（Secure Multi-Party Computation），MPC 由姚期智在1982 年提出，指参与者在不泄露各自隐私数据情况下，利用隐私数据参与保密计算，共同完成某项计算任务。

   该技术能够满足人们利用隐私数据进行保密计算的需求，有效解决数据的**“保密性”和“共享性”**之间的矛盾。多方安全计算包括多个技术分支，目前，在MPC 领域，主要用到的是技术是**秘密共享、不经意传输、混淆电路、同态加密、零知识证明**等关键技术，你可以认为多方安全计算是一堆协议集。

   - **秘密共享**：秘密共享的思想是将秘密以适当的方式拆分，拆分后的每一个份额由不同的参与者管理，单个参与者无法恢复秘密信息，只有若干个参与者一同协作才能恢复秘密消息。
   - **同态加密**：一种允许在加密之后的密文上直接进行计算，且计算结果解密后和明文的计算结果一致的加密算法。
   - **不经意传输**：不经意传输是一种可保护隐私的双方通信协议，消息发送者从一些待发送的消息中发送某一条给接收者，但并不知道接收者具体收到了哪一条消息。不经意传输协议是一个两方安全计算协议，协议使得接收方除选取的内容外，无法获取剩余数据，并且发送方也无从知道被选取的内容。
   - **零知识证明**：零知识证明指的是证明者能够在不向验证者提供任何有用信息的情况下，使验证者相信某个论断是正确的。允许证明者 prover、验证者 verifier 证明某项提议的真实，却不必泄露除了「提议是真实的」之外的任何信息。
   - **混淆电路**：混淆电路是双方进行安全计算的布尔电路。混淆电路将计算电路中的每个门都加密并打乱，确保加密计算的过程中不会对外泄露计算的原始数据和中间数据。双方根据各自的输入依次进行计算，解密方可得到最终的正确结果，但无法得到除结果以外的其他信息，从而实现双方的安全计算。

   对隐私计算了解多少?

   答：隐私计算是指在提供隐私保护的前提下，实现数据价值挖掘的技术体系。面对数据计算的参与方或意图窃取信息的攻击者，隐私保护计算技术能够实现数据处于加密状态或非透明状态下的计算，以达到各参与方隐私保护的目的。隐私保护计算并不是一种单一的技术，它是一套包含人工智能、密码学、数据科学等众多领域交叉融合的跨学科技术体系。隐私保护计算能够保证满足数据隐私安全的基础上，实现数据“价值”和“知识”的流动与共享， 真正做到“数据可用不可见。”

   隐私保护计算的目标是在完成计算任务的基础上，实现数据计算过程和数据计算结果的隐私保护。数据计算过程的隐私保护指参与方在整个计算过程中难以得到除计算结果以外的额外信息，数据计算结果的隐私保护指参与方难以基于计算结果逆推原始输入数据和隐私信息。

   对差分隐私了解多少?

   差分隐私(Differential Privacy)是Dwork[3] 在2006年针对数据库的隐私泄露问题提出的一种新的隐私定义。主要是通过使用随机噪声来确保，查询请求公开可见信息的结果，并不会泄露个体的隐私信息，即提供一种当从统计数据库查询时，最大化数据查询的准确性，同时最大限度减少识别其记录的机会，简单来说，就**是保留统计学特征的前提下去除个体特征以保护用户隐私**。

2. 对门限签名了解多少？主要的实现方案是什么?

   门限签名也称多重签名，m of n 门限签名，即系统中共有n个用户，任意m个用户聚在一起可以恢复出secret，任意少于m个用户聚在一起都不能恢复出secret。

   聚合签名在区块链中主要是用来实现交易的批量验证。

   门限签名技术在区块链中是降低拜占庭类共识算法通信复杂度的一种方式。

   在门限签名中：

```
1. 各方独立生成一个key。
1. 然后各方一起合作制作金库的锁。制作的过程为，各方基于各自的key来生成锁的一部分，最终合成为a single modular lock。
1. 解锁的过程为，各方依次进行解锁，每个密钥可以解锁一部分，在使用不同密钥进行一轮解锁后，达到设定个数后，锁被完全解开。
```

主要的实现方案包括：

- 基于RSA的门限签名方案
- 基于ECDSA的门限签名方案
- 基于Schnorr的门限签名方案
- 基于BLS的门限签名方案

3. 编写智能合约会用到哪些库？Openzeppelin、Wispper等仓库？智能合约静态代码编译过程？静态代码分析工具有哪些?
    1. Solidity标准库：Solidity是以太坊智能合约语言，其标准库提供了一系列常用的数据类型和函数，例如数学库、字符串库、时间库等等。
    2. OpenZeppelin库：OpenZeppelin是一个以太坊的开源项目，提供了一些常用的安全合约库，例如SafeMath数学库、AccessControl访问控制库、ERC20/ERC721代币合约等等。
    3. Web3.js库：Web3.js是一个以太坊的JavaScript库，提供了一些与以太坊交互的API，例如合约部署、调用合约函数、获取账户信息等等。
    4. Truffle框架：Truffle是一个以太坊智能合约的开发框架，它内置了Solidity编译器、测试框架、合约部署工具等等，提供了便捷的合约开发、测试和部署工具。
    5. Remix IDE：Remix是一个基于浏览器的Solidity集成开发环境（IDE），它提供了编写、编译、部署和测试Solidity合约的功能，同时内置了一些常用的合约模板和库。
4. DeFi会用到哪些智能合约库?

    DeFi应用通常需要使用多个智能合约库来实现各种功能。以下是一些常用的智能合约库：

    1. OpenZeppelin：一个开源的智能合约库，提供了安全、经过测试的智能合约，可以用于构建各种 DeFi 应用。
    2. Compound Protocol：Compound 是一个去中心化的借贷协议，它的智能合约库提供了一组用于实现去中心化借贷协议的智能合约。
    3. Uniswap V2 Core：Uniswap 是一个流动性协议，它的智能合约库提供了一组用于实现去中心化交易的智能合约。
    4. Aave Protocol：Aave 是一个去中心化借贷协议，它的智能合约库提供了一组用于实现去中心化借贷协议的智能合约。
    5. Chainlink：Chainlink 是一个为智能合约提供外部数据的解决方案，它的智能合约库提供了一组用于实现安全、可靠的数据访问的智能合约。
    6. Gnosis Safe：Gnosis Safe 是一个开源的多签名钱包，它的智能合约库提供了一组用于实现多签名钱包的智能合约。
    7. MakerDAO：MakerDAO 是一个去中心化稳定币协议，它的智能合约库提供了一组用于实现去中心化稳定币协议的智能合约。

这些智能合约库可以通过 Solidity 或其他智能合约编程语言编写，并且大多数都是开源的，可以免费使用和修改。

5. 智能合约静态编译过程是如何实现的

   智能合约静态编译是指在智能合约代码被部署到区块链网络之前，将其源代码转换为可在虚拟机中执行的字节码。这个过程包括以下几个步骤：

   1. 语法解析和词法分析：将源代码解析成令牌（tokens）和语法树（syntax tree）。
   2. 语义分析：对语法树进行分析，检查代码的语法和语义是否正确，包括变量声明、类型检查、函数调用等。
   3. 中间代码生成：将语义分析后的代码转换成中间代码，中间代码通常是一种与源代码和目标机器无关的中间表示形式，比如LLVM IR。
   4. 优化：对中间代码进行优化，提高代码的执行效率。
   5. 目标代码生成：将优化后的中间代码转换成目标机器的机器码或者虚拟机指令。
   6. 打包：将目标代码和其他必要的文件打包成一个可执行的文件。

   静态编译的好处是在合约执行时可以更快地执行代码，因为编译后的代码是机器码或虚拟机指令，可以直接在虚拟机中执行，而不需要解释执行。同时，静态编译还可以减少在合约执行过程中消耗的计算资源和能源，从而降低成本。

   智能合约静态编译可以使用多种编译器和工具实现，比如Solc、Truffle等。这些工具可以将源代码编译成目标机器的机器码或虚拟机指令，并将其打包成可执行文件。在部署合约时，只需要将打包好的文件上传到区块链网络上即可。

6. 智能合约静态代码分析工具有哪些

   智能合约静态代码分析工具可以帮助开发人员和审计人员检测合约代码中的漏洞和安全问题。常见的智能合约静态代码分析工具包括：

   1. Mythril：Mythril 是一款针对 Solidity 智能合约的开源静态分析工具。它可以检测出包括重入攻击、溢出、调用可用性问题等多种漏洞。
   2. Slither：Slither 是一款专为 Solidity 语言开发的静态代码分析工具，能够检测出一些潜在的漏洞，例如可重入漏洞、未经检查的发送漏洞等。
   3. Oyente：Oyente 是一款 Solidity 合约的智能分析工具，能够检测出多种漏洞类型，例如不良的授权和可重入漏洞。
   4. Securify：Securify 是一款智能合约静态分析工具，可以检测出多种类型的漏洞，包括逻辑错误、未经授权的访问和恶意代码注入等。
   5. SmartCheck：SmartCheck 是一款针对 Solidity 合约的静态代码分析工具，能够检测出多种类型的漏洞，例如整数溢出、重入漏洞、可撤销函数漏洞等。

   这些工具通常使用静态代码分析技术，对智能合约代码进行分析和检测，从而帮助开发者和审计人员发现潜在的漏洞和安全问题，提高智能合约的安全性和可靠性。

7. 使用Golang时使用到哪些微服务架构?

   在使用Golang时，常见的微服务架构包括：

   1. Go Micro：Go Micro 是一个基于 Go 语言的微服务框架，它提供了简单的服务发现、负载均衡和消息传递等功能，可以快速构建高可用性的微服务。

2. gRPC：gRPC 是由 Google 推出的高性能 RPC 框架，使用协议缓冲区（Protocol Buffers）进行数据编码，支持多种编程语言，包括 Golang。
3. KrakenD：KrakenD 是一个轻量级的 API 网关，可以将多个后端服务封装为单个 API，支持动态配置和路由。
4. Moleculer：Moleculer 是一个分布式微服务框架，支持多种传输协议和服务发现机制，可以快速构建可扩展的微服务应用。
5. Gin：Gin 是一个轻量级的 Web 框架，可以用于构建 RESTful API 和 Web 应用程序，具有高性能和低内存占用的特点。
6. Buffalo：Buffalo 是一个基于 Golang 的 Web 开发框架，它提供了强大的命令行工具和插件系统，可以快速创建 Web 应用程序。

这些微服务架构都可以帮助开发者更快速、高效地构建可扩展的微服务应用程序。