session 5

PPT 内容

Modern SNARKs consist of two components: an information-theoretic interactive oracle proof (IOP) [1]; and a compatible cryptographic commitment scheme, which “compiles" the IOP into an argument system [5].

> 现代 SNARKs 由两个部分组成：
> ① 信息论**交互式**预言机证明(oracle proof) (IOP)；
> ② 兼容的密码承诺(`commitment`) 方案，它将 IOP “编译” 成一个论证系统(argument system)

This note introduces several commonly used commitment schemes.

> 本节主要介绍几种常用的承诺 `commitment` 方案。

An IOP is “information-theoretic" in that it provides soundness and zero-knowledge guarantees even when the prover and verifier are computationally unbounded. To make this possible, the proof system makes the idealised assumption of “oracle access": in other words, the verifier can only access the prover’s messages through random queries.

> 1. IOP是基于 `信息论` 的安全性证明，不依赖于计算机算力的假设。即使证明者和验证者可以执行无限量的计算，也无法 欺骗 或者 泄露 私密信息。
> 为此，证明系统需要做出了“预言机访问” 的理想化假设：换句话说， 验证者只能通过随机查询访问证明者的消息。

The commitment scheme instantiates this oracle access using cryptographic primitives (e.g. a one-way function): as a consequence, the resulting argument system is only secure with respect to a computationally bounded prover and/or verifier. To realise a succinct argument system, the chosen commitment scheme must provide low communication complexity relative to the computation being proven.

> 2. 承诺(commitment) 方案使用密码原语 (`cryptographic primitives`) （例如单向函数, ex: `Hash()` ） 实例化此 oracle 访问：因此，由此产生的论证系统仅对于计算有界的证明者和/或验证者而言是安全的。 为了实现简洁的论证系统，所选择的承诺方案必须提供相对于被证明的计算而言较低的通信复杂性。

理解 `commitment`

密码学中的 "commitment" 可以让发送方在不透露具体内容的情况下，**预先向接收方承诺一些信息**。具体来说，commitment 可以让发送方将某个数值、消息或其他数据“锁定”在一个不可更改的状态中，并将这个状态发送给接收方。接收方在稍后的某个时候，可以通过对该状态的验证，确信发送方没有对该信息进行修改。

一个 `commitment` 通常包含两个步骤：`commitment` 阶段和 `reveal` 阶段。

- 在 `commitment` 阶段，发送方会生成一个 `commitment` (定长字符串)，可以用来表示某些数据或信息，但是发送方不会透露实际的数据或信息。
- 在 `reveal` 阶段，**发送方透露实际的数据或信息**，并且接收方可以使用之前的 `commitment` 来验证这些数据或信息是否与 `commitment` 匹配。

在日常生活中，commitment 的一种类比是写下一个答案，并将它封存在一个信封里，然后将信封交给另一个人。这个人在需要检查答案时，可以打开信封并验证答案是否正确。

一些应用 commitmen 的场景可能包括：

- 在博弈中，承诺将要采取的行动或策略，以防止对手从承诺中推断出任何信息。
- 在在线交易中，承诺某些机密信息（如交易金额、地址等），以确保信息在交易完成前保密。
- 在密钥协商中，双方可以通过 commitment 预先承诺一个随机数，并在后续的密钥生成中使用它，以确保密钥的安全性。

Pedersen commitment

Commitment Schema 主要目的是提供 **“针对双方的信息互换方案”**，通过隐藏关键值来避免信息泄露。
Pedersen Commitment 也不例外，但不同于普通的承诺方案，其可以通过不揭露相关信息来达到确认双方**是否拥有某一个相同的值**

举两个例子：

> 1. 小诸和小周都想在赤壁之战使用“火攻”，但彼此不清楚对方是否知道“火攻”这两字信息，他们又不希望“火攻”被曹操知道。为了确认他们彼此都有这项信息，他们进行了 Pedersen Commitment 确认彼此都使用相同的战术——火攻, 同时又不会被曹操方面知道这个火攻策略

> 2. 假设 Alice 给 Bob 转账金额 n， 发送和接收方是知道明文（转账的金额）的。 Pedersen Commitment 想要保证参与双方知道（才能生成正确的承诺），但是其他人不知道的情况下，能验证明文（转账的金额）是 Alice 所承诺的明文。

构建 Pedersen commitment 如下:

1. Setup:

receiver chooses...

- Large primes `p` and `q` such that `q` divides `p-1` : 大素数 `p` 和 `q` 且使得 `q` (能) 除以 `p-1`
- Generator `g` of the `order-q` subgroup of Z_p* : $\mathbb{Z}_p = \{0, 1, 2, \ldots, p-1\}$ 的 `q` 阶子群的生成元 `g`
- Random secret `a` from Z_q
- $h = g^a \ mod \ p$
  - Values `p, q, g, h` are public, `a` is secret

2. Commit

to commit to some message `m`, sender chooses random r∈Zq and sends

$$c = g^m \, h^r \quad mod \quad p$$

to receiver

This is simply

$$g^m (g^a)^r = g^{m+ar} \quad mod \quad p$$

3. Reveal

to open the commitment, sender reveals `m` and `r`, receiver verifies that

$$c = g^m \, h^r$$

> Reference: https://course.ece.cmu.edu/~ece734/fall2015/recitations/7-18734R-Commitments.pdf

---

例 2

针对下面这个问题：

> 假设 Alice 给 Bob 转账金额 n， 发送和接收方是知道明文（转账的金额）的。 Pedersen Commitment 想要保证参与双方知道（才能生成正确的承诺），但是其他人不知道的情况下，能验证明文（转账的金额）是 Alice 所承诺的明文。

假设 Alice 给 Bob 转账金额 `n` , 定义 pedersen 承诺如下：

$$C = n \cdot G + r \cdot H$$

其中 `n` 是转账金额（想承诺的数值, 类似于上例中的 `m` ）， `r` 是盲化因子（即随机数），G 和 H 都是椭圆曲线上的公开点。很显然，$n \cdot G$ 和 $r \cdot H$ 都是该椭圆曲线上，以 n 和和 r 为私钥的公钥，知道 $r \cdot G$ 难以计算出反推 r。

那么，Alice 生成自己的承诺如下：

$$C_{Alice} = n \cdot G + r_{Alice} \cdot H$$

> r_{Alice} 是 Alice 选择的盲化因子。

Bob 类似生成承诺：

$$C_{Bob} = n \cdot G + r_{Bob} \cdot H$$

双方公布给对方自己的承诺 `C` ，然后双方分别相减：

$$\begin{aligned} C_{Alice} - C_{Bob} = C_{Alice} &= n \cdot G + r_{Alice} - n \cdot G - r_{Bob} \cdot H \\ &= (r_{Alice} - r_{Bob}) \cdot H \qquad \text{(前提是双方真的有相同的 } n \text{)} \end{aligned}$$

双方都公布这个差值，如果完全相等，那么就能保证 n 是相等的且不会被外人知道。
反之如果差值是不相等，也就代表着双方有一方没有拥有正确的信息。

`r` 的隐藏

在密码学的 Pedersen commitment 中,既然 m 会在 reveal 阶段被透露给 Verifier, 那么 Prover 想要隐藏的信息是什么呢？

Prover 怎样揭示 m 而又不透露 m 的真是信息呢？ 这是怎么做到的？ 请用公式说明

在 commitment 中，既然 `reveal` 阶段会把 `r` 发送过去，一开始在 commitment 阶段隐藏 r 的意义是什么呢？请举例说明

在密码学的 commitment 中，隐藏随机值 r 的目的是为了防止选择性披露攻击（selective disclosure attack）和承诺撤销攻击（commitment revocation attack）。

- 比如 `Pedersen`(波塞冬) `commitment` 方案, 该方案使用离散对数问题来保护承诺值的安全性。在 commitment 阶段，Alice选择两个随机数r和s，并计算出承诺值 $C = g^r * h^s$（其中 g 和 h 是两个预先约定的生成元）。然后，Alice 将 C 发送给 Bob。在 reveal 阶段，Alice 会将 r 和 s 发送给 Bob，以供 Bob 验证 C 的正确性。在该方案中，隐藏 r 的目的在于防止 Bob 在 commitment 阶段推断出 s 的值，并从而推断出 C 的值。

- 因此，隐藏 `r` 可以增强协议的安全性，防止欺诈行为。

- 在 Pedersen 承诺方案中，如果r和s都暴露了，那么攻击者可以通过计算 $C = g^r * h^s$ 来推断出承诺值 C 的值。因此，在 commitment 阶段中隐藏 r 的目的在于防止攻击者推断出承诺值 C 的值。

- 然而，在 reveal 阶段中，Alice 会揭示 r 和 s 的值，从而允许Bob验证承诺值的正确性。一旦Bob获得了r和s的值，他可以计算出 $C' = g^r * h^{s'}$ ，其中 s' 是由 Alice 在 reveal 阶段揭示的 s 的值。然后，Bob 可以将 C 和 C' 进行比较，以验证 C 是否等于 C'，从而验证原始数字的正确性。

- 因此，在 Pedersen承诺方案中，隐藏r的目的在于保护承诺值的安全性，而在 reveal 阶段中揭示 r 和 s 的值，则是为了允许验证者验证承诺值的正确性。

- 在承诺撤销攻击中，攻击者可以声称他们知道随机值 r，但是承诺者声称他们不知道，然后承诺者会选择另一个随机值 r'，从而揭示出原来隐藏的信息。如果随机值 r 是公开的，攻击者可以轻松地声称他们知道这个值，从而对承诺者进行欺诈。通过隐藏随机值 r，承诺者可以防止攻击者进行这种欺诈行为。

因此，隐藏随机值`r`是承诺的一部分，可以帮助确保承诺的机密性和完整性。在 reveal 阶段，承诺者公开随机值`r`和实际值，以便验证承诺的完整性。

why hash 不能代替 Pedeson

在密码存储中使用的密码哈希函数、消息认证码、数字签名等。

尽管在某些情况下，哈希函数可以用于构造承诺方案，例如使用Merkle树的承诺方案，但是使用哈希函数来替代Pedersen承诺是不可取的，因为哈希函数无法提供Pedersen承诺所需的隐藏属性，也无法保证隐藏承诺值的离散对数问题的安全性。

hash function-based commitments have some limitations compared to Pedersen commitments. One limitation is that hash functions are **not homomorphic(不是同态的)**, which means that it is not possible to add or subtract hash function outputs in a meaningful way(这意味着不可能以有意义的方式加减哈希函数的输出) .

In contrast, Pedersen commitments are homomorphic and allow for linear combinations of commitments to be computed efficiently(允许有效地计算 commitments 的线性组合。).

Additionally, hash functions are vulnerable to preimage attacks(**原像攻击**), whereas Pedersen commitments are computationally binding and hiding assuming the discrete logarithm problem is hard in the underlying group.

因此，尽管哈希函数在某些情况下可以用于构造承诺方案，但是在需要承诺的情况下，Pedersen承诺是更安全和更适合的选择。

2. Hash commitment

`Hash commitment` 是一种基于哈希函数的 commitment 方案，它可以表示任何长度的数据或信息。假设我们有一个数据 m，我们可以使用以下公式生成 Hash commitment：

$$c = H(m||r)$$

其中，H 是一个哈希函数，$||$ 表示拼接操作，r 是一个随机数。发送方会把 c 发送给接收方，同时保留 r 作为私有信息。在 reveal 阶段，发送方透露 r 和 m ，接收方可以验证：

$$c' = H(m||r)$$

如果 $c = c'$，则表示 commitment 验证通过。

这里只是给出了两个简单的例子，实际上在密码学中还有很多其他类型的commitment方案，如RSA commitment、Merkle commitment等。这些方案的公式可能会更加复杂，但是它们都遵循commitment阶段和reveal阶段的基本原则。

IOP Versus IP

信息理论交互式 Oracle Proof（IOP）和交互式证明（Interactive proof）不完全相同：

- 交互式证明（Interactive proof）是指一个交互式的证明协议，其中一个证明者（称为证明者）试图向另一个实体（称为验证者）证明某个命题的真实性，通常是一个数学定理或某个计算问题的解。证明者通过在交互过程中向验证者发送一系列消息（例如数字、文本或计算结果），并基于验证者返回的响应来构建他们的证明。交互式证明在计算复杂性理论和密码学中都有广泛的应用。
- 信息理论交互式 Oracle 证明（IOP）则是一种特殊类型的交互式证明，其中证明者和验证者都与一个称为 Interactive Oracle 的第三方实体交互。Interactive Oracle 是一个黑盒程序，可以回答一些特定的查询，例如回答某个输入字符串的某个位置上的字符是什么(安比实验室 intro 第三课被绑架的天使 😇)。在 IOP 中，证明者和验证者都可以向 Interactive Oracle发送查询，并基于 Oracle 的响应来构建他们的证明

因此，IOP是交互式证明的一种特殊形式，其中引入了一个额外的理想化实体（交互式 Oracle）来协助证明过程。

在单变量多项式承诺中，假设有一个多项式 $f(x)$ 和一个随机数 r。发送方可以通过将多项式 $f(x)$ 和随机数 r 作为输入，计算出一个承诺 C，并将其发送给接收方。承诺 C 可以被视为多项式 $f(x)$ 和随机数 r 的哈希值，它可以被安全地公开和存储，而不会泄漏原始多项式的信息。

接收方可以在不知道多项式 $f(x)$ 和随机数 r 的情况下，验证承诺 C 的有效性。如果发送方想要公开多项式 $f(x)$ 和随机数 r，接收方可以通过验证承诺 C 和公开的多项式 $f(x)$ 和随机数 r 来检查它们是否相同，从而验证多项式的完整性和一致性。

单变量多项式承诺（Single-Variable Polynomial Commitment，以下简称PCommit）是一种密码学原语，旨在提供一种安全且可验证的方式来承诺某个单变量多项式的值，同时保护其隐私性和完整性。

假设Alice想要发送一个多项式 $f(x)$ 的承诺给Bob。在这个过程中，Alice不想让Bob知道多项式 $f(x)$ 的具体值，但她希望Bob可以在未来验证 $f(x)$ 的值确实是她在承诺中所声明的值。

PCommit 的基本思想是将多项式 $f(x)$ 转换成一个对称密码学哈希函数 H 的输入，并计算其哈希值 $c = H(f(x), r)$，其中 r 是一个随机数（称为“盐”或“随机因子”）。然后，Alice向Bob发送哈希值 c 作为多项式的承诺。

在这个过程中，Alice不需要向Bob泄露多项式 $f(x)$ 的具体值，因为哈希函数 H 是单向函数（即不可逆的）和碰撞抗性的，这意味着除非Bob找到 $f(x)$ 的具体值，否则他无法从哈希值 c 推导出 $f(x)$ 的任何信息。因此，Bob只能在未来验证 $f(x)$ 的值是否与哈希值 c 匹配，而不能从哈希值 c 中获得任何其他有关多项式 $f(x)$ 的信息。

为了进一步增强PCommit的安全性和可靠性，通常会添加一些附加条件，例如：

- 强制在承诺之前，多项式 $f(x)$ 必须满足某些特定的约束条件。
- 要求盐 r 必须是随机生成的，并且不能在多个承诺之间重复使用。
- 通过使用零知识证明（Zero-Knowledge Proof，以下简称ZKP）来证明Bob已经正确地验证了多项式 $f(x)$ 的值。

通过这些措施，PCommit可以提供强大的隐私保护和数据完整性，使得Bob可以在未来验证多项式 $f(x)$ 的值是否符合承诺，而无需知道多项式 $f(x)$ 的具体值。这使得PCommit在加密货币、零知识证明、区块链和分布式系统等领域中得到了广泛的应用。

在单变量多项式承诺中，假设 Alice 事先选择了一个多项式 $f(x)$ 并对其进行承诺，即将多项式 $f(x)$ 的某个点 x_0 的值 $f(x_0)$ 进行哈希或加密等操作得到承诺值 C。现在 BOB 想要验证这个承诺是否正确，即验证承诺值 C 是否对应于多项式 $f(x)$ 的值 $f(x_0)$。

下面介绍一种基于交互式的验证方法：

1. BOB 随机选择一个数 r 并发送给 Alice。
2. Alice 将多项式 $f(x)$ 和承诺值 C 发送给 BOB。
3. BOB 计算多项式 $f(r)$ 的值，并将计算结果 $y = f(r)$ 发送给 Alice。
4. Alice 检查 r 是否与承诺值 C 中的随机数相等，如果相等，则接受验证通过，否则拒绝验证。
5. Alice 将多项式 $f(x)$ 在点 r 处的值 $f(r)$ 计算出来，并将计算结果 $y = f(r)$ 发送给 BOB。
6. BOB 检查 y 是否与之前自己计算的 $y = f(r)$ 值相等，如果相等，则接受验证通过，否则拒绝验证。

如果 Alice 想要欺骗 BOB，即将一个错误的多项式或者错误的承诺值发送给 BOB，那么她需要预测 BOB 会选择哪个点 r 进行验证，并能够在该点处计算出错误的值来欺骗 BOB。但是，由于 BOB 可以随机选择 r，使得欺骗的概率很小。

这种方法的正确性基于多项式的插值定理，即一个 n 次多项式可以通过 $n+1$ 个不同点上的值唯一确定。因此，通过在不同的点上进行验证，可以保证承诺的正确性。

KZG

见 `KZG.pdf`