What did you learn?

> Advanced Protocol

Number 46: What do correctness, soundness and zero-knowledge mean in the context of a Sigma protocol?

Sigma protocols

Alice 想要给 Bob 证明她知道一些信息（即她知道的秘密），Sigma 协议就是用来干这个事儿的一些协议，一般形式如下： Alice 知道一个秘密；Alice 和 Bob 共享一些共同信息，then

1. Alice 给 Bob 发送一个值，称为承诺（commitment）；
2. Bob 随机选取一个挑战（challenge）并发送给 Alice；
3. Alice 计算一个响应（response）并发送给 Bob；
4. Bob 检查响应并决定接受（accept）或拒绝（reject）Alice 的声明。

> 传说中，如果把上面的过程画成一个图，这个图看起来像希腊大写字母 Sigma(Σ)，所以这个协议就叫Sigma协议

而从密码学角度来看，我们希望 Sigma 协议满足三个性质：

1. Correctness：Alice 知道秘密，Bob 接受 Alice 的声明；
2. Soundness：Alice 不知道秘密，Bob 拒绝 Alice 的声明；
3. Zero-knowledge：Bob 不知道秘密，Alice 不泄露秘密。

> 其实还有更正式的定义，但是给的链接已经 404 了

Defining a Sigma Protocol

令 k 是一个域(field)。我们重点研究一个线性函数 $f: W \to X$，它从一个 k 维空间（k - vector space）映射到另一个，其中 Alice 和 Bob 都知道一些公共的 $x \in X$。同时Alice也知道一个秘密 $w \in W$，使得 $f(w) = x$。Alice想要给Bob证明她知道 x 的原像。

> $f: W \to X$,Alice 知道 $(x, w) \in X \times W$ 使得 $f(w) = x$，而 Bob 知道 x。

针对这个结构的 Sigma 协议为：

1. Alice 选择一个随机数 r，令 $A = f(r)$，然后发送给 Bob（即 Alice 对 r 的承诺）；
2. Bob 随机选取 $c \in k$，发送给 Alice（此为挑战）；
3. Alice 计算 $s = r + c \cdot w$，然后发送给 Bob（此为响应）；
4. 如果 $f(s) = A + c \cdot x$，则 Bob 接受 Alice 的声明。

Sigma Protocol in elliptic curve

> 这节是按照自己理解拆分出来的，和原文顺序不同

现在很多密码学都是在椭圆曲线上完成的。而椭圆曲线是由形如 $P = (x, y)$ 的点和一个“无穷远”的特殊点构成的一个群。这些点满足某些方程，而曲线上两个点相加可以得到另外一个点，这个加法就是定义群的加法。我

们通常从一个大素数 p 开始，在基域 $k=\mathbb{F}_p$ 上计算，并考虑 $E_p=E\cap\mathbb{F}_p\times\mathbb{F}_p$ 中的点 P。

许多椭圆曲线协议是从使用点乘法生成密钥对开始的：所有人协商一个公共基点 P，然后每个人都可以选择一个密钥 $x\in\mathbb{F}_p$，随之计算相关的公钥 $Y=x\cdot P$。如果 Alice 想要注册她的公钥，注册公司肯定需要她证明她知道 x（不然随随便便一个人就能注册了），但是 Alice 又不想透露密钥的信息，Sigma 协议就是用来干这个事儿的。

这样就一一对应起来了，$W=\mathbb{F}_p, X=E_p$，f 就是点乘法，即 $f:W\to X, w\mapsto w\cdot P$ （这个函数是线性函数，“矩阵乘法”也是如此）。

> 按理说到这步已经 over 了，下文应该是给出另外一种角度

假如 Alice 的私钥是 x，公钥是 $Y=x\cdot P$，同时有人给她发送用其加密的 ElGamal 密文 (C,D)，Alice 可以解密然后证明她解密成功。也就证明了其确实掌握秘密 x。 一种方法就是计算一个解密共享（decryption share）$S=x\cdot C$；解密后的明文就是 $D-S$，但是任何人都可以干这步，因此还不能结束。 所以 Alice 想要证明其确实掌握满足 $Y=x\cdot P, S=x\cdot C$ 的 x。于是乎，我们可以设置
$W=k, X=E_p\times E_p, f(x)=(x\cdot P, x\cdot C)$ 。

Correctness

几乎所有协议中的正确性都意味着：如果每个人乖乖遵守协议，那么协议就会发挥作用。所以在 Sigma 协议中，正确性意味着：如果 Alice 知道 w 使得 $f(w)=x$，并且双方乖乖按照协议办事，那么 Bob 会接受 Alice 的声明。

Soundness

可靠性意味着 Alice 不能证明一个错误的声明。在 Sigma 协议中，Soundness 意味着：如果 Alice 不知道 w 使得 $f(w)=x$，那么 Bob 会拒绝 Alice 的声明。

> 原文的描述让我明白了垃圾的叙述对一篇文章影响是多么大
>
> Soundness means that Alice cannot prove a false statement. This trips up a lot of people because the first protocol they see is Schnorr's protocol for proving that $\boldsymbol{y=x\cdot P}$, so Alice is proving that such an $\boldsymbol{x}$ exists. But that is obvious! (Alice is also proving that she knows $\boldsymbol{x}$, which is more interesting but that's another property.) But let's look at the other example, Alice proving that $\boldsymbol{S}$ is a correct decryption share for $\boldsymbol{C}$ under public key $\boldsymbol{Y}$. Here, Alice is proving that an $\boldsymbol{x}$ exists such that $\boldsymbol{Y=x\cdot P}$ and $\boldsymbol{S=x\cdot C}$ which is not true for all tuples $\boldsymbol{(P,C,Y,S)}$. What's actually going on is that the image of $\boldsymbol{f}$ is a one-dimensional subspace of the two-dimensional $\boldsymbol{k}$- vector space $\boldsymbol{X}$. In our formalism, soundness means that Bob does not accept (except with perhaps negligible probability) unless $\boldsymbol{x}$ is in the image of $\boldsymbol{f}$ (that is, a preimage $\boldsymbol{w}$ exists such that $\boldsymbol{x=f(w)}$).

Sigma 协议还具有一种称为“special soundness”的性质。 非正式的说：如果 Alice 刚刚把她的承诺 A 发送给 Bob，则对于 Bob 发送过来的 c，Alice 找到可以让 Bob 满足的 r 值的概率有多大呢？假设只存在这么一个值，那么概率就是 $1/|k|$（小到可以忽略不计）。 特殊稳健性说的是，如果Alice即使只应对 $|k|$ 个可能的挑战中的两个，也能说服Bob，且前像 w 一定存在： 假设在挑战 c 时，爱丽丝会回答 s，而在挑战 $c'\neq c$ 时，爱丽丝会回答 s'，鲍勃会接受这两个挑战。然后设 $d=(c-c')^{-1}$，我们可以这样做，因为 k 是一个域，而 $c\neq c'$，再用一下线性代数就可以知道 $w=d\cdot(s-s')$ 。

> 我认为这里有两层意思：一是 Alice 不要随便证明；二是如果一个攻击者能够在给定一组合法的证明之后生成出另一组合法的证明，那么该攻击者就能够有效地欺骗系统，从而破坏系统的安全性。这意味着系统中的证明必须是唯一的，且不能被伪造。

Zero-knowledge

Bob 很高兴，因为协议是可靠稳健的。但是 Alice 想要 Bob 不能从协议中学到 w 的任何信息，其实 Alice 想要的更多，比如在与 Bob 运行完协议后，Bob 也应该不能向 Charlie 证明 他知道 Alice 的秘密。

> 而零知识证明包含的东西就更多了

Sigma 协议是零知识的证明.....并不存在！与人们在课本零知识一章中学习Sigma协议后可能会猜测的相反，Sigma协议通常不是零知识，而密码学的初学者在考试时最好记住这一点。

> 不过它满足了一个较弱的要求，称为诚实验证者零知识（honest-verifier zero-knowledge）。

然而，在零知识的背景下讨论 Sigma 协议并不是完全没有意义的:人们可以通过几种方式使它们成为零知识，其中最实用的方式就是使其成为非交互式协议。但这是下章的话题...

Number 47: What is the Fiat-Shamir transform?

> 看看如何把 Sigma 协议变成零知识的

Sigma 协议可以让 Alice 向 Bob 证明一些事情，快速而使实用，但是要求双方同时在线。但是很遗憾，这个协议不是零知识的，而仅仅只是 honest-verifier zero-knowledge.

What is the Fiat-Shamir transform?

The Fiat-Shamir transform 是可以将 Sigma 协议转换为非交互式证明的方法。这个方法是由 Fiat 和 Shamir 在 1986 年提出的。这就意味着 Alice 可以直接向 Bob 发送邮件就行了，而不需要实时在线；而 Bob 也不用立刻就读，也无需向 Alice 发送挑战。 它还能让 Alice 把任何 Sigma 协议变成数字签名方案，从而断言 "知道这个 Sigma 协议秘密的人签署了该信息"。

> "someone who knows the secret for this Sigma protocol has signed that message"

ALice 只需创建一次签名，然后将其发布到公告板上，每个看到签名消息的人都可以检查签名，而无需联系 Alice 本人。零知识自然而然也就来了，因为 Bob 等人没必要再做任何事情了。

> Fiat-Shamir transform 本身有点争议，有人说可以追溯到更早

Sigma 能够通过四个算法（Commit, Challenge, Respond, Check）来实现：

```
Alice                                  Bob
-----                                  -----
co,st = Commit(secret,public)
         ---------- co --------->
                                       c = Challenge()
         <--------- c  ----------
r = Respond(st,c)
         ---------- r  --------->
                                           Check(co,c,r)
```

而在 Fiat-Shamir transform 中，Alice 会选择一个哈希函数 H，并用其创建自己的挑战：

```
Alice                                World
-----                                -----

co, st = Commit(secret,public)
c = H(public,co)
r = Respond(st,c)
         ------ co,r ----------->
                                     c = H(public,co)
                                     Check(co,c,r)
```

> 本质思想就是不想让 Alice 随便控制挑战内容，之前是 Bob 选择挑战，现在是 Alice 自己用哈希函数生成挑战（这两个虽然是不同人完成的，但是均不受 Alice 控制）

如果想签名一个消息，Alice 会将其添加到哈希函数中并发送消息：

```
Alice                                World
-----                                -----
c = H(public,co,m)
r = Respond(st,c)
 ------ m,co,r ----------->
```

如果 H 是一个随机函数，那么挑战显然是均匀随机的，与爱丽丝的公开信息和承诺无关。安全性分析中认为 Alice 仅能把 H 当作一个预言机，而不能直接访问其代码。在这种情况下，Alice 在不遵循协议的情况下做出正确响应的概率(特别是当她不知道秘密时)与 H 的范围的大小成反比。而这个概率不用多说了，小到可以忽略。

这种安全性分析的方法叫做随机预言机模型（random oracle model，ROM）。有人会告诉你这么分析是有严重缺陷的，因为有一个人为的反例：即在 ROM 中安全的方案，对于任何实际的哈希函数 H 来说都是不安全的。这个反例表明如果你努力去完成一个愚蠢的方案，那么你就会得到一个愚蠢的方案（？？？？？？他要不要听听他在说什么？）

> there's an artificial counter- example of a scheme that is secure in the ROM but is insecure for any actual hash function H. What this counter-example shows is that if you go to enough effort to make a stupid scheme, you can end up with a stupid scheme. 感觉他意思就是：人类做不出真正的随机，所以，以能做到随机为前提，再怎么分析出花儿来都毫无意义

事实上， Fiat-Shamir transform 从 1986 年提出至今，依然完好无损，无人提出一个可行的攻击方案来针对 Fiat-Shamir 转换后的 Sigma 协议方案。

Number 48: What is the purpose and use of a TPM?

> **T**rusted**P**latform**M**odule（可信平台模块） 原文连这个解释都没有

我们首先理解一下 TPM 设计出来是为了克服什么问题？ 答：信任（Trust）问题 信任什么？ 我们知道运行在电脑上的内存和软件，它们可以通过操作系统直接访问到。所以如果密钥等秘密消息直接存储在内存中，并能被软件访问，那么如果一个攻击者获得操作系统级别的访问权限，那么他就可以获得所有秘密信息（比如直接从存储密钥的内存位置直接读取密钥）。

有一个解决方案是令密钥永远不能直接存储在可以被软件访问到的内存里。但是！安全应用必须用到密钥，所以密钥必须能够以被使用的姿态出现，怎么做呢？ 一种方法是通过利用一个软件无法访问到的密钥来对内存中的密钥进行封装：例如，在一个单独的硬件上刻录密钥，并能利用该密钥执行某些加密操作，软件可以利用存储在硬件上的密钥进行各种操作，如将密钥包装后存储在内存中，但永远无法直接访问这个密钥。

上述就是 TPM 作用的概述。 一个 TPM 有一个 RSA 密钥对，被称为 存储根密钥（SRK, Storage Root Key），私钥部分完全保密。1）其它软件的密钥可以由 SRK 的私钥封装加密（这个过程被称为“绑定”（“binding”））；2）除了简单的封装，TPM 还可以将封装密钥与某些平台测量值（platform measurement，也可以翻译成平台度量？）绑定，只有当这些平台测量值与创建密钥时的值相同时，才能解除对这类密钥的封装。这个过程叫做密封（sealing）；3）TPM 还可用于生成加密密钥和；4）执行其他加密任务，其中一项任务被称为远程验证（remote attestation），它可以创建硬件和软件配置的哈希密钥摘要，允许第三方验证软件是否被更改过。

> 这就是 use of a TPM

这里真正需要理解的是，通过将安全性下沉到硬件级别，并确保将其交给一个独立的硬件，该硬件有自己的固件和电路，无法从外部进行更改，这样系统就不会暴露到软件漏洞下，因此更值得信赖。 所以 purpose of a TPM 归结起来就是：克服信任软件完全可靠的问题。

> To overcome the problem of trusting (or rather not trusting) software to be completely reliable.

Number 49: Describe the basic ideas behind IPSec and TLS.

Internet Protocol Security (IPsec) 和 Transport Layer Security (TLS) 都旨在为不安全网络环境中的双方建立一条安全信道。 一般来说，通信双方都使用某种机制来建立私人会话密钥（预共享或密钥协商协议），并使用对称密钥加密技术完成大部分的通信。还有一些关于身份验证的细节，本章略过。虽然这两个在最终目的上有相似之处，但是它们在实现上有很大的不同。

IPSec 位于 OSI 模型的网络层（network layer），旨在为两个终端提供完整性（integrity）、真实性（authenticity）和保密性（confidentiality）。由于它位于网络层，因此会对来自上层的数据进行盲目加密、MAC 和打包，然后再向下发送。这实际上在两个终端之间创建了一个虚拟网络链接，而无需确保终端应用程序已适当保护数据安全。这通常用于企业 VPN 解决方案，因为它是远程访问企业网络的快速解决方案。但缺点是，连接一旦建立，就很难限制应用程序使用该连接。

而 TLS 则是在 OSI 模型的应用层（application layer）建立安全连接。我们看到，TLS 被大量用于确保 HTTPS、STARTTLS 等网络协议的安全，因此，每个连接/应用程序都将独立协商/建立安全连接。从安全的角度来看，这是非常有吸引力的，因为单个受损的通道不应该对其余通道产生任何影响。虽然 TLS 可以被视为一种更灵活的方法，但对于两个节点之间的大量连接而言，它确实会比 IPSec 产生一些开销。

> 自己总结：IPSec主要用于保护网络通信的安全（目标宏大而数量稀少），而TLS主要用于保护应用程序通信的安全（数量众多且单体简单），所以IPsec也会更复杂一些

Number 50: What is the BLS pairing-based signature scheme?

这周看个签名方案：BLS pairing-based signature scheme

这个方案使用了椭圆曲线上的 Weil pairing，本质上是曲线上阶数除以 n 的点上的**双线性**形式（用乘法符号表示），取值为 n 次统一根。

> 看不懂思密达 essentially a bilinear form (with multiplicative notation) on points of order dividing n on a curve, taking values in n^{th} roots of unity.

假设有一条椭圆曲线 $E/\mathbb{F}_{3^l}$，按照原论文中的符号进行计算。方案如下

KeyGen: 令 $E/\mathbb{F}_{3^l}$ 为一条椭圆曲线，q 是这条曲线阶数的最大素因子，令 P 为线上一点且阶数为 q，并随机选取 $x \in \mathbb{Z}_q^*$，最后计算 $R = x \cdot P$，则公钥为 (l, q, P, R)，私钥为 x。

Sign: 为了签名消息 $M \in 0.1^*$，我们将 M 映射到群 $\langle P \rangle$ 上的一个点 P_M，这步参考文献（正文第一行）的3.3节（如下），本质上是个哈希函数。令 $S_M = x \cdot P_M$，签名 σ 为点 S_M 的 x 坐标，且 $\sigma \in \mathbb{F}_{3^l}$。

> ***MapToGroup***$_{h'}$**:** The algorithm defines $h : \{0,1\}^* \to G^*$ as follows:
> 1. Given $M \in \{0,1\}^*$, set $i \leftarrow 0$;
> 2. Set $(x, b) \leftarrow h'(i \parallel M) \in \mathbb{F}_{p^l} \times \{0,1\}$;
> 3. If $f(x)$ is a quadratic residue in $\mathbb{F}_{p^l}$ then do:
>
> 3a. Let $y_0, y_1 \in \mathbb{F}_{p^l}$ be the two square roots of $f(x)$. We use $b \in \{0,1\}$ to choose between these roots. View y_0, y_1 as polynomials of degree $l-1$ over $\mathbb{F}_p$. Then ensure that the constant term of y_0 is not greater than the constant term of y_1 when viewed as integers in $[0, p]$ (swapping y_0 and y_1 if necessary). Set $\tilde{P}_M \in E/\mathbb{F}_{p^l}$ to be the point $\tilde{P}_M = (x, y_b)$.
>
> 3b. Compute $P_M = (m/q)\tilde{P}_M$. Then P_M is in G.
> If P_M is in G^* then output $MapToGroup_{h'}(M) = P_M$ and stop.
>
> 4. Otherwise, increment i, and goto Step 2; If i reaches 2^I, report failure.

{: width="600px"}

Verifiy: 给定公钥 (l, q, P, R)，消息 M 和签名 σ:

- 在阶数为 q 的曲线上找到一个点 S，其 x 坐标为 σ，y 坐标属于 $\mathbb{F}_{3^l}$。如果不存在这样的点，则拒绝该签名。
- 设 $u = e(P, \phi(S))$，$v = e(R, \phi(h(M)))$，其中 e 是曲线上的魏尔配对，ϕ 是一个 $E \leftarrow E$的同态，h 就是上面提到的。
- 如果 $u = v$ 或 $u^{-1} = v$ 则认为签名有效，否则拒绝签名

> 两个等式分别对应满足条件的 y 和 $-y$

Number 51: What is the security model for ID-based encryption, and describe one IBE scheme.

ID-based encryption

在公钥密码体制中，如果 Alice 想给 Bob 发送消息，她需要 Bob 的公钥，而这往往是一个非常长的比特串。 而在实际中，Alice 更想使用 Bob 的个人公开信息（如电子邮件地址或者名字）来加密消息，这样既不需要获取和记忆很长的字符串，而且也不需要验证其真实性（至少我们可以确定肯定不是 Charlie 的公钥）。为了实现这一点，我们就需要 ID-based encryption（IBE，基于身份的加密）。

在 IBE 中，有一个实体叫做 私钥生成器（Private Key Generator，PKG），当 Bob 向 PKG 验证身份以后，他就可以拿他的 ID（例如电子邮箱）申请私钥，而 PKG 会根据 Bob 的ID 和主密钥计算出其私钥。这样 Alice 就可以使用 Bob 的 ID 来加密消息，而 Bob 可以使用私钥解密。 但是这会产生一个问题，通过主密钥，PKG 可以对任何消息进行解密，这叫做密钥托管（key escrow），意思就是说你必须认为 PKG 不会读取你的消息或者你不

在乎 PKG 会读取你的消息。在公司中，高级管理者一般都有权限读取下级的邮件，因此 IBE 方案在这种情况下是很恰当的。

一个 IBE 方案应该包含四个算法：

- **Setup**：输入安全参数，并输出（秘密）主密钥和（公开）系统参数，如信息和密文空间。
- **Extract**：输入 ID 和主密钥，并返回与 ID 相对应的私钥
- **Encrypt**：输入明文和 ID，并返回密文。
- **Decrypt**：输入密文和私钥，并返回明文。

Security model

Boneh 和 Franklin 在 2003 年提出一种 IBE 方案。他们证明，在类似于 CDH 问题很难解决的假设下，他们的方案在随机预言机模型中是 IND-ID-CCA 安全的。这意味着（假设所有哈希函数都是随机预言机），任何 PPT 攻击者，在以下安全游戏中获胜概率都不会大于 1/2：

> any attacker, running in polynomial-time with respect to the security parameter, wins the following security game with probability that is only negligibly (with respect to the security parameter) more than 1/2:

首先，攻击者可以：

- 请求任意 ID 对应的私钥
- 请求解密任意 ID 下的任意密码文本。

然后，攻击者选择两个消息 m_0 和 m_1，一个从未请求过私钥的 ID ${ID}^，然后攻击者会收到一个密文*c*，其由*{ID}*^加密m_b而得到（b$ 是随机选择的）

接着，攻击者可以：

- 请求与 ID^* 之外任何 ID 对应的私钥
- 请求解密除 (c, ID^*) 之外任何 ID 下的任何密文

最后，攻击者输出一个比特 b'，如果 $b' = b$ 则攻击者获胜。

An IBE scheme

Boneh 和 Franklin 所给出的方案依赖于非退化 (non-degenerate) 双线性映射 $e : G_1 \times G_1 \to G_2$，其中 G_1（我们将运算写为加法） 和 G_2（我们将运算写为乘法） 是阶数为质数 q 的群。他们用 Weil pairing 来实例化构建了方案，细节可以参考原文。重要的是双线性（bilinearity）：$e(aP, bQ) = e(P, Q)^{ab}$。

方案大致如下： 主密钥为某个非零 $s\in\mathbb{Z}*q*，$ID$对应的私钥为*sH(ID)*，*H*是一个哈希函数，它将比特串映射为*G_1*的元素。有两个公共参数*P,P*{pub}=s\cdot P\in G_1。为了加密消息m，选择一个随机字符串\sigma并将m与\sigma的哈希值进行异或，从而创建c_m。然后m和\sigma一起哈希得到一个非零元素r\in\mathbb{Z}*q*。接着计算*e(H(ID),P*{pub})^r，将其哈希并与\sigma异或，得到密文c_{ID}。三元组(rP,c_{ID},c_m)$ 就是密文。

解密： 私钥 $d = sH(ID)$ 与密文 $(U, V, W) = (rP, c_{ID}, c_m)$ 首先计算 $e(d, U)$

> 根据双线性：$e(d, U) = e(sH(ID), rP) = e(H(ID), s \cdot rP) = e(H(ID), P_{pub})^r$

将其哈希并与 V 异或，得到 σ； 然后将 W 与 σ 的哈希异或，得到 m。 为了验证，我们需要将 σ 和 m 哈希在一起看是否能得到 r，从而使 $U = rP$。

Number 52: Pick an advanced application concept such as e-Voting, Auctions or Multi-Party Computation. What are the rough security requirements of such a system?

在密码学中考虑的不仅是对遵守规则的玩家的安全，还有对**不遵守规则的玩家**的安全。让我们从投票、拍卖和多方计算的角度来研究这个问题。

> 感觉这章内容很适合用于给小白普及密码学

What we mean by these three applications?

Voting

在投票中，投票者根据某种投票方案（得票最多者当选制（first-past-the-post）、替代投票制（alternative-vote）、赞同投票制（approval voting）或其他）选择候选人。投票应该保密，仅仅合法的投票者可以进行投票，每个候选人只能投一票，投票必须是有效的(例如必须投一个真正的候选人)，最终结果必须是正确的，且选民不能被强迫等等，诸如此类的安全需求不胜枚举。

Auctions

对于拍卖，我们可能希望竞价是不公开的，我们可能不信任拍卖师（auctioneer），可能有多个物品参与拍卖，有多个可能的最终价格，获胜出价/价格的选择将取决于某种算法，最终输出可能需要可审计。

> 出价人是 bidder；拍卖师，拍卖商是 auctioneer

Multi-Party Computation

对于多方计算（我们指的是对一组各方的私人输入进行函数计算）来说，安全要求比较简单，我们只希望函数的输出被公开，而不希望输入被公开（根据输出可以计算出的除外）。不过，虽然这个目标比较简单，但其功能却比拍卖和投票更广泛，因为我们要求任何函数都应能够计算。

What makes these operations interesting?

最大的特点就是坏人将会是协议的一部分。 之前的加解密操作中，Alice 和 Bob 都是诚实的，坏蛋是协议外偷听的人。而在投票，拍卖和多方计算中，我们不可以相信任何人，坏人可以是想要多投几票的投票者，可能是想篡改结果的计票人，也可能是一个试图中标但是出价又不是最高的人，还可能是一个试图计算出未中标出价的拍卖师！

刚刚说的虽然在搞事，但起码还是按照协议走的，无非是试图多走几步或者获得点额外信息。但是还有一类坏人，甚至不按规则行事，即不遵守协议。他们可能发送压根儿就是错误的信息，但这些信息 "看起来 "是有效的，但随后会对协议产生错误的结果。我们需要防范这种所谓的 "恶意 "行为。

刚刚说的只是一个一个的坏人，但是在现实中，坏人甚至会拉帮结伙，他们会合作，他们会互相帮助，联合起来攻击这个系统。我们需要确定在我们的协议中可以容忍多大的坏人联盟。例如在 MPC 中，诚实占多数和不诚实占多数的情况有很大不同。对于诚实多数协议，我们可以确保诚实的各方最终总是得到有效的函数输出。而对于不诚实多数协议，我们无法阻止不诚实的一方终止每个人的协议。

> （最后一句没太懂）For dishonest majority protocols we cannot stop a dishonest party from terminating the protocol for everyone.

还有一种情况： 我们需要防止出现谁先谁后的问题。这在学术中被称为 "公平性"（fairness）。例如，假设我们有一次选举，有三个投票人：A、B 和 C。假设选票是加密的，玩家 C 可能通过复制 A 的选票来确保 A 支持的候选人获胜（从而找出 A 投给了谁）。应防止这种情况发生。

还有一个概念： 在协议开始时，对手可以控制一组参与方，这就是所谓的静态对手（static adversary）；或者，随着协议的进行，敌手会决定它想贿赂哪些参与方，这就是所谓的自适应对手（adaptive adversary）。

我们可以看到，在这种高级协议中，人们可能会有大量安全方面的顾虑，也确实会有许多安全方面的结果。事实上，每个应用领域都可能需要不同的安全属性。鉴于可能的应用协议种类繁多，这意味着密码学需要解决的问题永无止境；因此！

密码学博士生需要解决的问题也永无止境。