我讲解的论文题目是：《隐私毯子：差分隐私置乱模型的隐私放大效应与求和方法》

OK, I'm going to be talking about Amplification and Summation in the Shuffle model of differential privacy.

在讲解我们的具体工作之前 我们需要首先介绍差分隐私和置乱模型

So, I shall start by introducing differential privacy and the shuffle model before talking about exactly what we've done in this work.

在差分隐私中 我们有一个数据集 数据集中包含用户的数据

OK, so in differential privacy, we have a database, which has people's data in it.

我们想对数据集进行数据分析

And we're going to perform some analysis.

我们希望数据分析的输出结果不过多透露出数据集中某一个个体的数据

And the property that we want is that the output shouldn't tell you too much about any one person in the database.

为了形式化描述这一点 想象我们有另一个不包含我 而是包含艾蒙信息的数据集

So, in order to formalize this, we imagine we have some other database, which instead of having me in it, has Elmo in it.

我们不希望攻击者能通过观察输出结果 知道是我、还是艾蒙包含在数据集中

And we don't want the adversary to be able to tell whether it was me or Elmo in the dataset from looking at the output.

也就是说 攻击者无法区分这两个数据集的输出结果

And so, the adversary should not be able to distinguish these two outputs.

当然了 如果在可忽略的统计距离下定义统计安全性

Now, of course, if this was statistically secure with some negligible statistical distance,

则根据三角不等式

then by some triangle inequality argument,

即使两个数据集中所有的数据都不相同

you can see that even if every entry in the database was different,

你也不能区分两个数据集的输出结果

you wouldn't be able to tell any difference in the outputs,

这意味着输出结果无法告知你任何与数据相关的有用信息

and thus the output wouldn't be telling you anything about the data.

这种隐私定义也就没什么意义了

So, that would be useless.

因此 我们需要定义区分两个数据集输出结果的程度 不能让不可区分性过于严苛

So, we're going to have to relax our notion of what it means to be able to distinguish two datasets.

而这个定义就是差分隐私

And this definition is the definition of differential privacy.

差分隐私的定义是什么呢？

So, what's this saying?

我们有数据集x和x’ 两个数据集中只有一个数据项是不相同的

We've got x and x’. They are just two databases that differ in one place.

我们想讨论的是 数据分析过程给出某一输出结果的概率

And then, we're talking about the probability that any event happens on the output of the analysis.

大家可以把最后的δ看成是统计距离 即密码学安全常数定义下的小参数

You should think of this δ on the end is some cryptographically small parameter, like the statistical distance.

如果认为δ=0 这个约束条件描述的是

So, if you think of that being 0 for a second, the condition becomes just saying that

数据集x和观察数据集x’下得到相同输出结果的似然比不超过e^ε

the likelihood ratio between the case when you're looking at database x and when you're looking at database x’ is bounded by e^ε.

换句话说 攻击者无法获得、或以非常高的置信率获得过多某个个体的信息

So, this somehow says that the adversary can't learn too much about anyone person or learn anything with too much confidence.

我们如何使用这一定义呢？

OK, so how are we going to apply this definition?

最开始的时候 差分隐私是在可信模型下定义的

Well, originally the model that differential privacy was developed for is the trusted curator model,

我们有一个可信第三方 所有人都将数据提供给这个可信第三方

where we have some trusted third party, everyone gives all of data to this trusted third party,

可信第三方对数据进行统计 得到统计结果 并在结果上增加一些随机量

the trusted third party computes some statistics on that data with some randomness added to them,

最后把结果发送给分析方 也就是发送给攻击者

and then releases them to an analyzer who is also our adversary.

我们希望分析方能从分析结果中得到一些有用的信息…

And we want the analyzer to then be able to say something useful about…

数据的分析结果是可用的 但因为M满足差分隐私 分析结果不会过多透露个体信息

So, this analysis of the data is going to be useful, but without being able to tell too much about any one person, because M is differentially private.

随后 人们又提出了差分隐私的本地模型

Then there's the local model of differential privacy.

此模型下 我们不再拥有可信第三方了

So here, we avoid having a trusted third party.

每个个体都在本地对自己的数据进行随机化处理

We just say everyone is going to locally randomize their own data.

每个个体都在自己的数据上执行本地随机算法R

So, each person applies a local randomizer R to their data.

发送给分析方的结果数据本身就已经满足差分隐私性了

And the resulting message they send to the analyzer is already differentially private.

很明显 这意味着个体无法告知分析方太多与自己数据相关的信息

Now clearly, this means that the you can't tell the analyzer too much about your data point.

每一个个体传输给分析方的信息量都是有限的

There's a limit to how much data you can get across to the analyzer, how much information.

但如果收集到很多用户的信息 分析方仍然可以得到与群体相关的一些信息

But across a large number of users, the analyzer might still be able to say something useful about the entirety of the population as a whole.

这里我们就需要进行权衡了

So, there's a trade-off here.

很明显 与可信模型相比 本地模型不存在过多的信任关系

Obviously, we don't require as much trust if we're in the local model.

但本地模型的数据分析输出结果可用性较低

But we also can't get as much utility, and did as the classical result.

在可信模型下 对[0,1]下的实数值求和 引入的随机量为O(1)

Now in the curator model, you can get O(1) guarantee when doing summation of real numbers in [0,1].

但在本地模型下 引入的随机量不可能低于O(√n)

But in the local model, you cannot hope for better in O(√n) error.

简单来说 这是因为每个各地都需要在输入中增加O(1)的噪声

Roughly speaking, this is because each person has to add O(1) noise to their input.

把这些噪声加在一起 噪声和的方差为O(n) 标准差为O(√n)

And then, when you add them all up, you're going to get variance O(n). And the standard error is O(√n).

置乱模型位于可信模型和本地模型之间

So, the shuffle model is an intermediate model between these two.

置乱模型的基本思想是 我们有一个可信第三方

And the idea here is that we're going to have a trusted third party,

但我们只相信这个可信第三方会诚实地置乱所有的输入

but we're only going to trust it to shuffle the inputs.

可信第三方会把所有的输入放到随机置换函数中

So, it's going to take in all the inputs, apply a random permutation,

函数把输入随机置换位置后 可信第三方再将结果发送给分析方

and then show the result of that random permutation to the analyzer.

在实际中如何实现这一安全模型框架呢？

How we implement this thing where…

我们不关注这个问题 在论文中我们也没有考虑这个问题

We don't care about. We ignore too in this paper.

这里有一些实现的建议

But there's some suggestions.

可以通过MPC实现 可以让第三方直接实现 也可以用混合网络实现

You could do this in MPC or a third party, or you could use a MixNet or something.

如果可以构建匿名信道 则匿名信道天生就是实现随机置乱模型的一种方法

If you have an anonymous channel, this kind of automatically gives you a shuffle up.

我们论文中的实际贡献是什么呢？

OK, so what are our actual contributions in this paper?

到目前为止 我所介绍的都是其他学者的工作

So far, everything I've talked about has been other people's work.

我们考虑取值范围为[0,1]的实数值求和问题

Well, here we look at the question of real summation of real numbers in [0,1].

我们证明了 单消息置乱模型的可用性比本地模型更好

And we show that in the one-message shuffle mode, you can do better than you can in the local model.

但单消息置乱模型的可用性不会比可信模型更好

But you can't do as well as you can in the curator model.

我们还证明了一个新的置乱隐私放大效应

And we also managed to prove a new amplification by shuffling result.

我们不是第一个证明置乱隐私放大效应的学者

This is not the first time that someone proved amplification by shuffling result.

但我们在特定的场景下 在之前结果的基础上证明出了新的结果

But we are able to prove at least in certain aspects on previous results.

来看看n个参与方下的求和问题

So, real n-party summation.

每个用户有一个[0,1]之间的实数 我们想计算所有实数的和

All I mean is each user has some number in [0,1]. And we want to estimate the sum of them.

与本地模型相比 Cheu等人之前的工作在置乱模型下对协议进行了些许优化

Previous work by Cheu et al., there is a slight improvement here in the local model.

优化协议为单消息模型 但未能证明协议的噪声标准差已达到最优

But it's not in the asymptotics, sorry, in the one-message model, but it's not present in asymptotics.

然而 他们还证明出 如果每个用户可以上传O(√n)比特长的消息

They are able to show, however, that if you have O(√n) 1 bit messages,

或者说每个用户向置乱方发送O(√n)次消息 置乱方对消息进行置乱处理

or each person is able to put O(√n) messages into the shuffler, and they all get shuffled together,

则协议的噪声标准差和可信模型一样

then you can achieve the error of the curator model.

我们考虑当用户只给置乱方发送一次消息时 协议究竟能优化到何种程度

So, we're looking at what happens in the case where you can only put one message in.

在这种情况下 我们证明了 如果用户可以只发送一次log(n)比特长的消息

And in that case, we're able to show that with just one message of log(n) bits,

则协议的噪声标准差可以达到O(n^(1/6))

you can achieve a standard error of O(n^(1/6)).

即协议的噪声方差可以达到O(n^(1/3))

So, this is a variance of O(n^(1/3)).

但这已经是最优结果了 我们不可能获得更小的噪声标准差了

But you can't do better than that. This is the best you can hope for.

我接下来将为大家证明准确性上界

OK, so I'm going to prove the upper bound.

因为时间关系 我不会为大家证明准确性下界

But I'm not going to prove the lower bound, because I won't have time.

但我会解释协议的工作原理 解释如何在单消息置乱模型下得到此种准确性结果

But I'd like to explain how this protocol work, how you can get this accuracy in the one-message shuffle model.

这是本地随机算法的执行过程 算法非常简单

So, this is our local randomizer. It's pretty simple.

此算法只需要做两件事情

And there are only really two things going on here.

当获得真实输入后 我们将其转换为固定精度下的值

You take your input. We're going to put it into fixed point precision.

也就是说 我们在k固定精度下对输入进行随机舍入处理

So, this is randomized rounding to some number in k precision.

随后 我们将执行k元随机答复协议

And then, after that, we're going to do k-ary randomize response,

也就是说 每个用户随机抛掷一枚有偏的硬币

which is to say that each person will flip a biased coin.

每个用户有γ的概率返回一个与输入独立、满足均匀随机分布的答复结果

And with probability γ, they will return a uniformly random answer independently of their input.

每个用户有1-γ的概率答复真实结果

And with probability 1-γ, they will just tell the truth.

最后 分析方将调用deBias算法 使统计结果满足无偏性

OK, and at the end, the analyzer is going to have to de-bias this.

这一步处理过程非常简单 只需要执行一次线性映射

But that's straightforward. It's just a linear map.

这就是本地随机算法的执行过程

OK, so this is our local randomizer.

为什么此算法可以实现我们给出的准确性要求呢？

So, why does it have the accuracy that I claim?

此算法包含两个噪声源

Well, there are two sources of error here.

第一个噪声源是将输入随机舍入到固定精度时引入的噪声

One is in the randomize rounding, when we map to the fix point.

此噪声的方差是O(n/k^2)

This incurs an error with variants O(n/k^2),

因为有n个参与方 所以有O(n)的噪声

O(n) because there are n parties,

而随机舍入又会引入方差为O(1/k^2)的噪声

O(1/k^2) because that's the variance of the randomize rounding.

第二个噪声源是一部分用户会给出与输入完全独立、满足均匀随机分布的答复结果

And then, there's the error due to the people who are lying, inputting uniform random response

一共有大约O(γn)用户会给出错误的答复

And there are O(γn) of them approximately.

每个用户错误答复所引入的噪声量为O(1)

And each one is going to be adding something with variance O(1).

因此这部分噪声源将引入方差为O(γn)的噪声

And so that contributes variance O(γn).

在接下来的几分钟我会为大家证明

And I'll show in a couple of minutes that

我们需要将γ设置为k/n乘以一个与δ和ε相关的函数

we're going to have to take γ to be k/n times some function of δ and ε.

如果把系数k/n带入到最上方的噪声量中 我们就会得到噪声方差为O(n/k^2)+O(k)

And if you sub that k/n in the top there, you can see that the variance comes out as O(n/k^2)+O(k).

因此 我们希望k大致等于O(n^(1/3))即可

And so, we want k to be about O(n^(1/3)).

如果令k等于O(n^(1/3)) 则噪声方差将变为O(n^(1/3))

And taking k to be O(n^(1/3)) gives a variance of O(n^(1/3)).

因此 噪声标准差为O(n^(1/6))

And therefore, we get this O(n^(1/6)).

这就是协议噪声标准差为O(n^(1/6))的原因

So, that's where this O(n^(1/6)) coming from as the standard error.

为什么我们要让γ等于这个值呢？为什么这么设置就够了呢？

OK, so why do I want to take γ to be this? Why does this suffice?

为了证明这一点 我们需要考察攻击者在此模型下可以得到哪些信息

In order to understand why this works, we're going to look at what the adversary sees.

这是攻击者的视角

So, this is the adversary's view.

攻击者能得到信息可以等价为一张直方图

And the adversary's view is equivalent to a histogram.

攻击者只能知道每一个输入值、或者说每一个原始值被上传了多少次

They just learned how many times each input has been submitted, the origin message has been submitted.

因此攻击者得到的是所有输入值的一张直方图

So, that's a histogram of the possible messages.

而攻击者知道 在得到这张直方图的过程中 有的用户在说谎 有的用户在说实话

And we know that these messages have come from some people lying and some people telling the truth.

我们把给出随机答复的用户所形成的直方图看作绿色直方图

So, those people who put random responses in, we can consider as this green histogram.

把给出真实答复的用户所形成的直方图看作红色直方图

And those people telling the truth, we consider as the red one.

攻击者明显不知道直方图中哪一部分是红色的、哪一部分是绿色的

The adversary obviously can't see which is which.

但我们要告诉攻击者直方图中哪一部分是红色的、哪一部分是绿色的

But we're going to tell the adversary which is which.

我们要给攻击者一个礼物 告诉攻击者哪些用户给出的是真实答复

We're going to give them a present, which basically tells them the set of parties that responded truthfully.

我们还假设攻击者知道除目标用户之外 所有用户的真实输入

And we're going to assume they also know what everyone else's input is except for yours.

这是差分隐私中的一个标准假设

And this is a standard assumption in differential privacy,

我们希望攻击者在足够多的背景知识下 仍然无法得到某个个体的信息

that we want to be able to protect against arbitrary side information.

有了这个礼物 攻击者可以从直方图中移除答复真实结果的用户

So, with this present, the adversary can never move everyone who told the truth data from the histogram.

如果目标用户给出的是真实答复 留给攻击者的就是绿色的直方图加上他的数据项

And they're left with the green histogram plus your data entry, assuming you told the truth.

如果目标用户在撒谎 他的答复就与真实数据相互独立 结果中不会泄露任何信息

If you lied, obviously your input is independent of your data, and so no information about you is going to be leaked.

因此 我们只需要担心当目标用户给出真实答复下的情况

So, we're only going to worry about the case where you told the truth.

我们把绿色直方图称为隐私毯子

OK, so this is the thing that we call the privacy blanket, this green histogram here.

隐私毯子的基本思想是 目标用户的数据可以被其他用户的独立随机数据所覆盖

And the idea is that your own piece of data is hiding amongst everyone else’s data.

隐私毯子可以盖住目标用户的真实数据

So, it kind of provides a cover for you to hide amongst.

攻击者需要有能力区分两种场景下的答复结果

So, these are the two situations that the adversary needs to be able to tell the difference between.

他需要告诉我们 答复结果是额外增加一个0得到的 还是额外增加一个1得到的

It needs to be able to tell whether or not there's been one extra 0 submitted or one extra 1 submitted.

我们需要考察两个概率分布的似然比

So, we're going to look at the likelihood ratio here.

正如我前面所提到的 差分隐私要求似然比不能大于e^ε

As I mentioned earlier, differential privacy roughly says the likelihood ratio isn't going to be greater than e^ε.

因此 我们只需要证明这两个直方图出现的似然比大于e^ε的概率最大为δ

So, it's suffice to show that the probability of that these two bins occur greater than equal to e^ε is at most δ.

为了证明这一点 我们需要计算似然比

OK, so in order to do that, we would need to work out this likelihood ratio.

当你的答复是0时 似然比就等于0提交的个数除以1提交的个数

So, in the case where you're submitting a 0, this likelihood ratio is just going to be the number of 0 submitted divided by the number of 1 submitted,

即似然比等于某个二项随机变量加1除以某个二项随机变量

which is some binomial random variable plus 1 divided by some binomial random variable.

我们要求两个随机变量的比值不大于e^ε

And this won't be greater than e^ε.

因为二项随机变量分布很集中于概率分布均值

Because binomial random variables are quite well concentrated about their means,

因此只要均值足够大 两个随机变量的比值就不会大于e^ε

so, so long as these means are large enough, we'll be fine.

均值应该为多大呢？

And how large the mean has to be?

均值应该等于ε和δ下的某个函数

It is going to just be some function of ε and δ.

如果想让均值等于某个常数 则可令γ等于k/n乘以某个ε和δ下的函数

And so, as we want the mean to be some constant, we can take γ to be k/n times some function of ε and δ.

如果你对差分隐私很熟悉 你可能就会对log(1/δ)/ε^2这个参数很熟悉

And if you're familiar with differential privacy, then you might recognize this log(1/δ)/ε^2,

这是在差分隐私中使用高斯机制时 所需要增加的噪声方差

this is the amount of variance you have to add when doing the Gaussian mechanism in differential privacy.

这个结果应该并不令人惊讶 因为隐藏信息用的二项随机变量近似等于高斯随机变量

So, this shouldn't be too surprising, because these binomial random variables that you're hiding amongst are approximately Gaussian.

这就是为什么会出现log(1/δ)/ε^2的原因

So, that's why it's log(1/δ)/ε^2.

这就是隐私性证明过程了

OK, so that's the privacy proof.

我们再来看看隐私放大效应

So now, what's this amplification thing?

我在前面已经证明我们可以用此协议实现实数求和

So, we've shown that we can do summation well.

这里的问题是 我们是否可以把相应的结论扩展到其它统计函数上？

And the question is can we extend this argument to other kinds of functionalities or other kinds of statistics you might want to compute?

Erlingsson等人最近证明出 如果你有一个满足一定条件的本地随机算法…

And Erlingsson et al. proved recently that if you have some local randomizer with…

假定本地模型下的差分隐私参数为ε\_0 其中ε\_0为一个常数

In the local model, this has ε\_0 differential privacy, where ε\_0 is at most some constant.

你就可以自动在置乱模型下满足差分隐私性 且对应的参数更优

Then, you automatically get differential privacy in the shuffle model with better parameters.

你需要引入一个非0参数δ 但参数ε\_0下会除以一个√n

You need to introduce a δ, which is nonzero, but you get this √n on the bottom here.

也就是说 如果n很大 则ε会降低很多

So, ε can come down by a lot, if n is large.

这是一个很有用的结论

And this is a useful result.

虽然我们希望获得较好的隐私性保证 但在很多场景下 我们不希望太好的隐私性保证

And if you're trying to prove high privacy, but in a lot of situations you don't want very high privacy,

我们希望的是合理隐私性保证和高可用性

what you want is moderate privacy and high utility.

我们扩展了Erlingsson的结论 可以让本地模型下的ε\_0设置得更大

So, we're able to extend the regime of this result to include larger values of ε\_0.

大家可以观察右侧的等式 ε的表达式中包含了一个e^ε\_0

You'll note that on the right here, there's an e^ε\_0 in our expression for ε.

这个参数会随着ε\_0的增大而快速增大

So, this does blow up reasonably quickly in ε\_0.

这里的关键点是 你可以令ε\_0等于O(log(n)) 但仍然能获得ε=1的差分隐私性保证

But the point is we can now take ε\_0 to be O(log(n)) and still get reasonable privacy guarantee of ε=1.

实际上 即使我们令ε\_0=1/2 令δ=10^(-6)

OK, and in fact, even in the case where we're taking ε\_0 to be 1/2, and δ to be 10^(-6),

这个图也告诉我们 我们结果中的隐私放大效应常数要比Erlingsson等人的结果更好

this graph shows that the constants that we're able to get out are better than those that come out of Erlingsson et al. result,

这是因为我们的攻击方法比他们的攻击方法更加直接

and kind of because we're attacking things more directly than they were.

我必须要说明的是 从系统和应用层面看 他们的结论要比我们的结论更通用

And I should mention though that their result is slightly more general than ours in terms of the systems in which you can apply it.

他们的模型和我们的模型有所不同

It works in models that are different to the one that we're looking at here.

因此 不能说我们得到了更强的结果

So, it's not quite as strictly stronger result.

你可能会问一个问题 如果我们应用置乱隐私放大效应

OK, so, then you might ask, if we have this amplification by shuffling,

是不是意味着我们可以寻找一个可以在本地模型下实现差分隐私的本地随机算法

can we do anything just by coming up with a local randomizer, same way of computing it in the local model,

然后直接在这上面应用置乱隐私放大效应？

and then applying amplification by shuffling to that?

可以说对 也可以说不对

And kind of yes, kind of no.

如果在本地模型下应用我这里给大家介绍的置乱模型下的随机算法

And the randomizer that I just showed you for use in the shuffle model,

则当随机算法的ε参数设置得很大 例如设置为O(log(n))时

it is a differentially private in the local model with some large value of ε, so O(log(n)).

直接应用置乱隐私放大效应 就可以得到我前面给出的隐私放大效应结果了

And you can recover the result that I just showed you directly by applying amplification by shuffling.

然而 如果你随便选择了一个本地模型下的本地随机算法

However, if you just take a local randomizer designed for use in the local model,

例如随便一个输入范围是[0,1]的随机算法

such as a randomizer running to [0,1],

比如简单随机答复 或者直接在结果上增加Laplace噪声

and randomize response, or adding Laplace random noise,

则置乱隐私放大效应的结果并不会太理想

then you won't do particularly well.

隐私放大效应的系数可能只能到√n 甚至只能到log(n)

You'll only get √n, or maybe a log factor improvement.

因此 你需要适当选择本地随机算法 使其在置乱模型下可以得到最优的准确性

So, you do need to choose your local randomizer so that it's optimized for accuracy in the shuffle model,

而不是选择一个在本地模型下最优的本地随机算法 再直接应用置乱隐私放大效应

rather than choosing it so it's optimized for the local model, and then just applying amplification by shuffling.

不好意思 点错了

Sorry.

这里还有另一个问题

So there's another question here,

我之前提到 我们的协议是在单消息模型下构建的 多消息模型下会得到什么结果呢？

which is I've said in the one-message model, we can do this, what about many-messages?

Chou等人证明 如果发送O(√n)个单比特消息 就可以进一步提高准确性

So, Chou et al. said that with O(√n) one-bit messages, you could do better.

置乱模型下的准确性结果可以和可信模型几乎完全相同

You could do as well as the curator model pretty much.

@注释：论文题目为《Differentially Private Summation with Multi-Message Shuffling》

在提交此论文后 我们最近在Arxiv上在线提交了一个笔记

And in a recent note which we've put online on the Archives since submitting this paper,

答案是一样的 的确可以做得更好

we already know that, yes, you can.

那篇笔记并没有涉及太多新的研究成果

And there's not a huge amount of new work in that note,

我们证明 可以让通信量降低到O(log(n)) 但是每个消息的长度也为O(log(n))

because it'll be shown that you can get away with O(log(n)) messages.

即发送O(log(n))个O(log(n))比特长的消息 而不是发送O(√n)个单比特长的消息

This is O(log(n)) messages or size O(log(n)), rather than O(√n) messages.

如何做到的呢？

And so, how do we do this?

协议的基本思想可以追溯到Ishai等人在2006年发表的论文

It basically boils down to some result by Ishai et al. from 2006,

那篇论文称 如果你有一个匿名通信信道 你就可以在统计安全下实现安全求和功能

which says that if you have anonymous channels, then you can do secure addition with statistical security.

我们可以在输入上加入随机噪声 每个参与方都可以在输入上加入随机噪声

So, we can add random noise to the inputs. Everyone can add random noise to their input.

随后 应用Ishai等人的协议 通过发送O(log(n))个消息实现隐私求和

and then apply this protocol by Ishai et al. with only O(log(n)) messages in order to get a private form of addition.

如果你可以接受发送O(log(n))次消息 你就可以得到与可信模型相同的准确性结果

So, there's no need to use… So, if you have O(log(n)) messages available, then you can get the accuracy of the curator model.

还剩下哪些公开问题呢？

So then, what questions are left open?

我前面讲到 发送O(log(n))个消息足以得到与可信模型相同的准确性结果

Well, I've said that O(log(n)) messages is suffice is to get the accuracy of the curator model.

单消息模型做不到那么高的准确性

And I've said that one-message doesn't.

你可能会问 如果是双消息呢？三消息呢？log(log(n))消息呢？

You might ask what if I have two-messages, or three-messages or log(log(n)) messages?

我们正在考虑这个问题 但到目前为止我们还不确定结果是什么

We're looking into this at the moment. But right now, we're not entirely sure.

我们还应该考察除求和以外的其它统计方法

But then, we should probably also be looking to do things other than addition.

这个模型的基本思想是 只要我们有一个置乱、或者说匿名通信信道

The motivation for this model is that we can maybe have a means of shuffling or anonymous channels.

我们就可以利用这个信道计算得到很多统计结果

And we can just use that to compute a lot of different things.

只需要一次性实现这个匿名通信信道

And so, we only have to implement this thing once.

我们就可以非常高效地计算很多不同的统计结果

And then you can compute a lot of different things with it very efficiently.

只有当可以利用这个模型计算很多不同的统计结果时 我们才能说这个模型是有用的

This makes sense only if you can actually compute a lot of different things with it.

因此 如果想说明这个模型是有用的 我们需要在此模型下构建其它统计计算方法

So, we need to be able to do something other than addition in this model if we're going to justify its use.

我们还需要解释如何实现置乱模型

We also need to explain how it's going to be implemented.

已经有相关的学者尝试在可信硬件下实现置乱模型了

And there's already work on people trying to implement shuffling in trusted hardware,

在差分隐私置乱模型出现之前 就已经出现了相关的工作

which came out before all this work on differential privacy in the shuffle model came out,

原因是 直观上看 在查看数据之前先置乱数据应该可以提高隐私性

and just because they thought it seemed intuitive that shuffling people's data before they looked at it would improve privacy.

我们或许可以使用MPC、混合网络、或者通过其它方法实现置乱模型

And we can also think of maybe doing this using MPC or using a MixNet or something along these lines.

我们需查看不同的实现方法 寻找最容易、最轻量级的实现方法来构建置乱模型

But these options need to be looked into and it has to be sufficiently cheap in order to justify using this as a model.

另一个问题是 置乱模型包含了一个可信假设

Another issue is there's a trust assumption here.

在证明过程中 大家可能已经注意到 我假设每个用户都会严格遵循协议要求执行协议

And in my proof, you might have noticed I assumed that everyone was following the protocol.

我们不需要假设每个用户都严格按照要求执行协议

And we don't really need to assume that everyone follows the protocol.

只要有一定比例的用户会按照要求执行协议 这个协议就能保证隐私性

It's sufficient for some positive fraction of people that we know in advance to follow the protocol.

但我们需要让足够多的用户按照要求执行协议

But we need there to be enough people following the protocol,

只要足够多的用户随机答复 攻击者就无法知道目标用户的回复结果是什么了

that enough of them respond randomly and don't tell the adversary what their response was.

因此 这些用户的安全假设并非是半可信的 他们必须是可信的

So, it can't be semi-honest. They have to be honest.

这样我们才能得到足够大的隐私毯子 从而隐藏目标用户的回复结果

And that there's enough of a privacy blanket for you to hide amongst.

你可能会想到 或许基于MPC的置乱协议可以帮助验证噪声是否已经正确添加

So, you might imagine that maybe some MPC means of shuffling will allow you to also verify that this noise is being added correctly,

噪声添加过程、或者随机答复过程也可以在MPC中进行 以保证噪声被正确添加

the noise, or the randomized response can be done inside MPC rather than being done outside of MPC,

这样就可以让这些用户的安全假设从可信退化到半可信

and that would remove this need to trust people.

如果上述问题都能实现 假设就都可以成立 整个系统就满足置乱模型了

And then, of course, so I've said the shuffle model, it will work if all of these things happen, if it can be implemented well if you can do functionalities in it.

如果我们能移除可信假设 协议会变得更好

If we can remove this trust assumption, then it's great.

不过 置乱模型可能也不是最优的

But maybe it's not the best functionality for this.

可能可以有更好、更容易实现的模型 可以支持更多的统计计算

Maybe there are functionalities which allow you to do more and are easier to implement.

因此 另一个问题是 是否还存在其它的模型 可以进一步优化统计计算的准确性？

And so, another question would be what are the single functionality could we have that would allow us to do a great variety of things cheaply.

这就是我的讲座内容 谢谢大家

OK, that's all I've got to say. Thank you.

感谢James的精彩讲座 大家有什么问题吗？

Thank you very much, James. Do we have any questions?

如果没有其它问题 那就让我们再次感谢James 以及感谢本分会场所有的演讲者

OK, if there are no further questions, let's thank James and all the speakers of the session again.