大家好

Hi, everyone.

感谢Stefan的介绍

And thanks Stefan for introductions.

我的名字是Ni 我是俄勒冈州立大学的博士研究生

And my name is Ni. I'm a PhD student at Oregon State University.

很高兴能来到这里介绍我们的工作

And I'm very happy to be here today to present our work.

我们的论文题目是：高效批不经意伪随机函数及其在隐私集合求交中的应用

This is, efficient batch oblivious pseudo-random functions with applications to private set intersection.

此工作是由我和Kalashnikov、Kumaresan、以及Rosulek共同完成的

And this is a joint work with Kalashnikov, Kumaresan, and also Rosulek.

在论文中 我们提出了一个高效的隐私集合求交协议

In this work, we proposed an efficient protocol that does apply private set intersection,

隐私集合求交是密码学中一个非常有趣的问题

which is an interesting problem in crypto.

我们以一个简单的场景为例 来看看什么是隐私集合求交

So, let's see a very simple scenario to see what is private set intersection.

例如 幻灯片上有两个参与方：Alice和Bob

So, for example, here we have two parties here Alice and Bob.

每个参与方都有一个集合 这里分别是X和Y

Each party has the set of items, here X and Y.

它们想计算两个集合的交集 但不想泄露其它额外的信息

And now, they want to compute the intersection of these sets, which does not reveal any additional information.

例如 Alice不能知道Bob集合中非交集的元素

So, for example, Alice doesn't know the rest of Bob’s items.

Bob也是类似的 他不能知道Alice集合中非交集的元素

And similar to Bob, he doesn't know the rest of Alice’s items.

这就是隐私集合求交问题的定义

So, there is a problem of private set intersection.

隐私集合求交的应用场景非常广泛

So, private set intersection has a lot of applications.

我这里给出的例子是通讯录匹配场景

And what I am showing in here is contact discovery.

例如 Alice有一个手机 里面存储着Alice的通讯录 Alice想要使用Skype

So, for example, we have Alice, she has a phone. She has her address book. And she wants to use Skype.

另一边 Bob是一个Skype服务器 里面存储着客户数据

On the other side, we have Bob, a Skype server provider with his customer data.

现在 Alice希望知道她的哪些朋友使用Skype 她希望使用Skype与朋友们聊天

And now, Alice wants to know which her friends use Skype so that she can chat with them.

很明显 两方希望计算集合的交集

So, clearly, they want to compute the intersection of the sets.

然而 Alice不想泄露自己的通讯录 因为这是她的个人信息

Yes, so, however, Alice doesn't want to reveal her address book, because this is her personal information.

Bob也面临类似的问题 他不能泄露自己的客户数据 因为这是客户的隐私

And similar to Bob, he doesn't want to reveal his customer data, because of customer privacy.

这个场景就需要使用隐私集合求交功能

So, we need privacy set intersection here.

当考虑隐私集合求交这个问题时 我们可能会提出下述解决方案

So, when we think about private set intersections, we might come up with following solutions.

Alice拥有集合X Bob拥有集合Y

Here Alice and Bob have assessed of X and Y.

他们分别对X中的元素求哈希 对Y中的元素求哈希

They simply hash its element of X, and its element of Y.

Bob随后将哈希值发送给Alice

Then Bob sends the hash value to Alice.

Alice对比两个集合的哈希值 并输出哈希值相等的元素 即交集元素

Alice simply compares two sets of hash values, and output whether it is intersection.

这个协议效率非常高 因为协议只涉及到哈希值的计算

So, this protocol is very fast, because we just need to compute the hash value, right?

协议涉及的通信量也很小

And they have low communications, yes?

但不幸的是 这个方案是不安全的 因为这个方案会泄露Bob输入集合的隐私

But unfortunately, they it is insecure, because it leaks the privacy of Bob’s inputs.

为什么呢？

So, why?

例如 X属于比较小的域 例如X为电话号码 只包含大约10个数字

So, for example, X comes from a small domain, like telephone numbers, about 10 numbers.

Alice直接计算上亿个电话号码的哈希值 并将结果与从Bob收到的结果对比

Alice simply hashes a billion elements, and then compares with the hash value just received from Bob.

这样 Alice就可以知道Bob的输入了

So, Alice can know Bob’s inputs.

这也是此协议被称为朴素哈希的根本原因

So, this is the reason why the protocol is called naïve.

因此这是一个不安全的PSI协议

However, it is an insecure PSI protocol.

为了解决这个问题 学者们提出了很多PSI协议

So, to handle the problems, several PSI protocols have been introduced.

当前最先进的PSI协议由Pinkas、Schneider、Segev和Zohner在2015年提出

And the state-of-the-art PSI protocol was proposed last year by Pinkas, Schneider, Segev, and Zohner.

隐私集合求交场景下的特殊情况为隐私相等性检测

And the special case of private set intersection is private equality test,

即两个参与方希望知道两个字符串是否相等

where two parties want to know whether 2 strings are equal.

他们的PSI协议通过不经意传输扩展实现隐私相等性检测

So, the current PSI protocol, they do private equality test using oblivious transfer extension.

他们还提出了一个高效的哈希技术 可以将隐私相等性检测高效转换为隐私集合求交

And they also proposed an efficient hashing techniques to efficiently transform private equality test into private set intersections.

我们的核心技术贡献是提高隐私相等性检测的效率

So, our main technique contribution is to improve private equality test.

我们来看看Pinkas、Schneider、Segev、Zohner提出的隐私相等性检测协议

So, let’s look at the current private equality test protocol of Pinkas, Schneider, Segev, and Zohner.

Alice拥有x 而Bob拥有y

So, Alice has x, and Bob has y.

我们想知道x=y是否成立

We want to know whether x=y, and nothing else.

幻灯片上给出了一个例子 x=001、y=011

So, here x=001, for example, and y=011.

他们协议的核心思想是对x和y进行逐比特对比

So, the main idea of their protocol is comparing bit by bit of x and y.

如何做到这一点呢？

So, how they can do it?

他们使用了一个安全的黑盒工具 此工具叫做不经意传输

They use a secure black box, called oblivious transfer.

Bob分别为0和1随机采样两个k比特长的字符串

So, Bob samples a random k-bit strings for 0 and for 1.

随后 Bob和Alice执行不经意传输协议 Alice的输入是她的第一个比特0

And now, Bob and Alice run oblivious transfer where Alice’s input is her first bit 0.

协议执行完毕后 Alice收到她第一个比特0所对应的字符串0

So, as the result, Alice receives a string indicating her first bit. This is string 0.

然而 Bob无法得知Alice在不经意传输中的输入是什么

And on the other side, Bob doesn't know anything about Alice’s inputs.

此性质由不经意传输协议的定义所保证

So, this is the definition of oblivious transfer.

他们继续对第二个比特、第三个比特执行此种操作

So, they do the same thing for the second and for the third bit.

现在 Bob从OT协议中取得自己输入所对应的字符串

So now, Bob takes a string from OT indicating for his input bits.

他的输入比特是011 因此他分别取得0、1、1所对应的三个字符串

Here, his input bit is 011. So, he takes a string for 0, for 1, and for 1.

他对几个字符串求异或 将结果发送给Alice

So, he xors them, and sends it to Alice.

Alice也取得OT执行完毕后的结果 即0、0、1所对应的三个字符串

Alice also takes strings from OT, see the received string 0, string 0 and string 1.

她对几个字符串求异或 并对比Bob发送过来的异或值

She xors them, and compare with the xored values just received from Bob.

这样Alice就知道双方的输入是否相等了

And Alice can output whether this is equal.

这就是当前最先进PSI协议的隐私相等检测过程

So, this is private equality test of the current PSI protocol.

协议执行过程中 双方逐比特对比x和y

And if we look at this protocol, they compare bit-by-bit of x and y.

而我们的基本思想是将2选1-OT协议替换为N选1-OT协议

And our idea is we want to replace several 1-out-of-2 OTs with only one 1-out-of-N OT.

当前的PSI协议使用了Kolesnikov和Kumaresan于2013年提出的OT扩展协议

So, and actually, the current PSI protocols, they use OT extensions. That's proposed by Kolesnikov and Kumaresan in 2013.

这意味着每轮协议只能同时对比8比特长的x和y 我后面会讲解为什么会这样

So, it means that they can compare only 8 bits of x and 8 bits of y. I can show you why they just do it.

因此 他们的PSI协议中N=2^8

And at the result, the N value equals to 2^8.

这意味着他们的协议会对x和y的比特长度提出要求

So, it means that their protocol still depends on the length of x, as well as the length of y.

他们的协议还需要通过多个OT实现隐私相等检测过程

And they also need several number of OTs to do private equality text.

在我们的工作红 我们提出了一个OT扩展协议 此协议中的N可以任意大

So, in our work, we propose OT extension that can work for any value of N.

在我们的协议中 N可以为无穷大

So, here we propose for an infinite number.

这样 我们只需要使用一次OT即可实现隐私相等检测过程

And as a result, we just need only one OT to do private equality test.

这里是Pinkas等人所提出协议的另一个观察结论

So, here is another observation for Pinkas’s protocol.

我们将此协议看成一个黑盒 如果Alice的输入是x 而Bob的输入是k

So, if we put this protocol in black box, and if Alice has the input x, and Bob has the input k,

这里k包含了6个数据块

here k is 6 blocks, this one,

在PSI协议执行完毕后 Alice收到字符串0、0、1的异或值

and now, after running the protocol, Alice receives the xor value of the string 001.

我们可以将此异或值看成F\_k(x)

So, we can say it is F\_k(x).

这意味着协议执行完毕后 Alice只能知道F\_k(x)

So, it means that Alice learns only F\_k(x).

而Bob已知全部的6个数据块 因此Bob可以对任意y计算得到F\_k(y)

And Bob knows 6 blocks. He can compute F\_k(y) for any y.

这里一个非常重要的性质是 如果x≠y 则对Alice来说F\_k(y)看起来是个随机数

So, the very important property here is if x≠y, F\_k(y) looks random to Alice.

这么看来 此协议本质上就是一个不经意伪随机函数

So, this is roughly oblivious pseudo-random function.

不经意伪随机函数是Freedman、Ishai、Pinkas和Reingold在2005年提出的概念

That was proposed by Freedman, Ishai, Pinkas, and Reingold in 2005.

进一步 如果Bob将异或值 也就是F\_k(y)发送给Alice

And moreover, if Bob sends F\_k(y), I mean xor values to Alice,

则Alice可以简单地比较F\_k(x)和F\_k(y) 得知这两个值是否相等

Alice simply compares F\_k(x) and F\_k(y). And she can output whether it's equal.

这就是隐私相等检测协议的原理

So, this is private equality test protocol.

在接下来的讲座中 我会聚焦于OT扩展协议本身

So, the rest of my talk focuses on oblivious transfer extensions.

我们将首先介绍原始的2选1-OT扩展协议 再介绍N选1-OT扩展协议

We will show an original 1-out-of-2 OT extension, and then a 1-out-of-N OT extension.

最后介绍我们构造出的∞选1-OT扩展协议

And our protocol is 1-out-of-∞ OT extension.

随后 我们会介绍∞选1-OT扩展协议和我们提出的OPRF实例之间的关系

Then, we will show the relationship between 1-out-of-∞ OT extension with our OPRF instance.

这里的OPRF对原始OPRF的定义进行了弱化

I mean here OPRF is the kind of relaxation of OPRF.

我后面会详细讲解定义具体的弱化点

And I will explain more about this one later.

我们将我们的OPRF应用在PSI上

And we apply our OPRF on PSI.

从而构造了一个半诚实安全的PSI协议 比当前PSI协议执行效率提高了3倍

At the result on semi-honest PSI, we have 3 times faster than the current PSI protocol.

正如我前面所说 当前PSI协议的执行效率依赖于集合元素的比特长度

And as I said, the current PSI protocol, they depend on the length of items.

我们的协议移除了协议执行效率与集合元素比特长度的依赖关系

So, for our paper, we remove the the dependence on the length of items.

我们先来简单介绍Beaver的OT扩展协议

So, let's go to Beaver’s oblivious transfer extension.

OT扩展协议的基本思想是 可以用少量OT和对称密码学操作构造大量的OT实例

So, the main idea here, we can do many OTs based on the few OTs and some symmetric key.

Beaver在数十年前首次提出了这个想法

And this idea is first proposed by Beaver about decades ago.

现在 我要向大家介绍一个非常著名、非常高效的OT扩展协议

And now, I tend to describe a very famous and efficient OT extension protocol.

此协议由Ishai、Kilian、Nissim和Petrank于2003年提出

That was introduced by Ishai, Kilian, Nissim and Petrank in 2003.

此协议的基本思想是 在基础OT协议中 Alice选择一个随机的k比特长字符串t^1

So, the main the idea is, in the base OT, Alice choose a random k-bit string t^1.

她计算得到另一个字符串t^1⊕r

She computes another string t^1⊕r.

在另一端还有个参与方Bob 他有一个单比特字符串s\_1

On the other side, we have Bob. He has one-bit string s\_1.

如果s\_1=0 则Bob接收到第一列字符串t^1

If s\_1=0, Bob receives the first column, here is t^1.

如果s\_1=1 则Bob接收到第二列字符串t^1⊕r

And if s\_1=1, Bob receives the second column, here is t^1⊕r.

也就是说 Bob可以根据自己的选择比特s\_1接收到字符串q\_1

So, it means that Bob will receive q^1, and the values depend on his choice bit s\_1.

OT协议定义的功能就是这样的

So, this is the oblivious transfer definition.

现在 他们重复上述过程k次 每次都使用相同的r

So now, they they repeat the previous step k times. And they use the same r.

最后 Bob收到包含k列的矩阵q

So, at the results, Bob receives k columns of q.

每列q\_i的值都由比特s\_i所决定

And the value of q\_i depends on the bit s\_i.

现在 他们使用轻量级的密码学技术伪随机数生成器将k个OT扩展为n个OT

So now, they use a very cheap crypto technique called pseudo-random generator to extend the length of OT to n.

这里n要远远大于k n大约等于2^20

And here, n is very larger k, n is about 2^20.

在整个协议执行过程中 Alice得到了矩阵T和矩阵T⊕r

So, I'm around to get the matrix T and the matrix T⊕r.

我们将这组矩阵称为OT矩阵

So, I'm saying here is the OT matrix.

如果我们从行的角度观察这个矩阵 并且令t\_i表示第一个矩阵的第i行

So now, if we look at the OT matrix by row, and if we say t\_i is a row in number i of the first matrix,

如果令r\_i表示r的第i个比特值 则第二个矩阵的第i行就等于t\_i 则此时q\_i=t\_i

and if r\_i is the bit number i of r, so a row of number i of the second matrix is equal to t\_i. And also q\_i is equal to t\_i.

左右两行下标相同 取值也相同

So, I'm saying here row number is the subscript.

如果r\_i=1 则第二个矩阵的第i行等于t\_i⊕1^k 此时q\_i=t\_i⊕s

So, if r\_i=1, the row number i of the second matrix is equal to t\_i⊕1^k. And q\_i=t\_i⊕s.

也就是说 r\_i或者等于0 或者等于1 长度为1比特

So, it means that r\_i comes from 0 or 1, 1 bit.

而q\_i=t\_i⊕(r\_i)^k⊙s

And q\_i is equal to t\_i⊕(r\_i)^k⊙s.

现在 我们把目标聚焦于OT矩阵本身

So now, we just focus on the rows of the OT matrix only.

如果Alice的输入是r\_i Bob的输入是s 则Alice已知t\_i 她可以计算t\_i的哈希值

And if now Alice’s input is r\_i, and Bob’s input is s, so now Alice knows t\_i, she can compute the hash of t\_i.

Bob收到了q\_i Bob可以计算q\_i的哈希值 以及q\_i⊕s的哈希值

Bob receives q\_i. Bob can compute the hash of q\_i, and the hash of q\_i⊕s.

这里一个非常重要的性质是 t\_i=q\_i 或者t\_i=q\_i⊕s 具体等于什么取决于r\_i

So, the very important property here is t\_i is equal to q\_i, or is equal to q\_i⊕s. The value depends on r\_i.

也就是说 H(q\_i)和H(q\_i⊕s)有且仅有一个值等于H(t\_i)

So now, it means that one of H(q\_i) or H(q\_i⊕s) is equal to H(t\_i).

而另一个值对Alice来说看起来像随机数

And the other value looks random to Alice.

换句话说 Bob有两个值 Alice只知道其中的一个值

So, in other words, Bob has two values, and Alice knows one of them.

在2013年 Kolesnikov和Kumaresan指出 IKNP协议中包含重复编码

So, in 2013, Kolesnikov and Kumaresan presented repetition encoding in IKNP protocol.

如果称C是一个k个比特值重复编码k次的编码过程 我们再来研究一下IKNP协议

And if we say C is a k-repetition encode from 1 bit to k bits, so here you are on the view of IKNP protocol.

第二个矩阵的第i行等于t\_i⊕C(r\_i)

The second row is equal to t\_i⊕C(r\_i).

而q\_i等于t\_i⊕C(r\_i)⊙s Bob

And q\_i is equal to t\_i⊕C(r\_i)⊙s.

Bob计算的是幻灯片上这两个值的哈希值

And Bob computes two hash values.

现在 如果把随机编码移除 将C替换为纠错编码

So now, instead of C as a repetition encoding, they replace it as a kind of error correcting code.

则此协议可以支持长度最大为8比特的选择比特值r\_i

So, their protocol can work for any i with the length up to 8 bits.

这也是为什么当前PSI协议只能对比8比特长的x和8比特长的y

So, that is reason why the current PSI protocol only compare 8 bits of x and 8 bits of y.

举例来说 我们令C的输入为3比特长

So, for example, here we have C from 3-bit length.

此时Bob需要根据幻灯片上的公式计算8个哈希值

So, Bob compute 8 hash values, but follow this formulas.

我们得到了一个非常重要的性质：H(q\_i⊕C(r’)⊙s)=H(t\_i⊕[C(r’)⊕C(r\_i)]⊙s)

So, we have the very important property here, H(q\_i⊕C(r’)⊙s)=H(t\_i⊕[C(r’)⊕C(r\_i)]⊙s).

现在 我们观察等式中的绿色部分

So now, we look at green colors here.

如果r\_i=r’ 则绿色部分等于0

If r\_i=r’, so this green color is becoming to 0.

绿色部分与s逐比特与 计算结果还是0

And when we bitwise-AND s, it becomes to 0.

这意味着我们得到的是H(t\_i)

And it means that we get H(t\_i).

换句话说 Bob可以计算m个哈希值 而Alice只能得到其中一个哈希值

So, in other words, Bob can compute m hash values, and Alice now one of them.

这也是为什么此协议被称为N选1-OT的原因

So, that is reason their protocol is called 1-out-of-N oblivious transfer.

从安全角度看 他们的协议要求C(r’)⊕C(r\_i)的汉明距离要大于k

So, for the securities, their protocol needs C(r’)⊕C(r\_i) has the Hamming weight more than k.

我们翻到下一页幻灯片 讲解其中的原因

So, let's go to the next slide to see the reason.

Bob能得到什么？

So, what does Bob have?

他计算得到H(q\_i⊕C(r’)⊙s)

He computes H(q\_i⊕C(r’)⊙s).

这个值等于H(t\_i⊕[C(r’)⊕C(r\_i)]⊙s)

So, it means it is equal to H(t\_i⊕[C(r’)⊕C(r\_i)]⊙s).

如果r\_i≠r’ 则此表达式计算得到的值对Alice来说是个随机数

If r\_i≠r’, we need this expression looks random to Alice.

然而 如果Alice可以通过某种方式猜测r’的值

So, we have some ways that Alice can guess r’.

如果Alice知道了r’ 她就知道了t\_i 她也就知道了r\_i

So, it means she knows r’, and she knows t\_i, she knows r\_i.

这实际上意味着Alice知道此表达式所有红色部分的取值 但不知道蓝色部分s的值

It means she knows red colors in this expression, except the blue colors, here is s.

如果C的最小汉明距离为k 则C(r’)⊕C(r\_i)的汉明距离大于k

So now, if we have C that has the minimum distance k, then C(r’)⊕C(r\_i) has hamming weight more than k.

这意味着Alice必须正确猜测s中的k个比特

So, it means that Alice must guess k bits of s.

这样一来 此协议的安全性就得到了证明

So now, this protocol achieves the security.

下面是我们的观察结论

So, here is our observation.

我们发现 我们不需要使用编码算法

We see that we don't need encoding.

我们将C替换为一个随机函数

So, we replace C as a kind of random functions.

这是一个非常小的技巧 但这个技巧让我们的协议变得非常厉害

So, it is a very small trick, however, it makes our protocol very powerful.

如果我们将C替换为随机函数 则我们的协议可以支持任意长度的r\_i

So, if we replace C as the random function, our protocol can work for any r\_i with any length.

也就是说 这里的r\_i可以为任意比特长

So here, r\_i can with any length.

从Bob的角度看 他可以计算得到H(q\_i⊕C(r’)⊙s)

So, on the Bob’s side, for any r’, he can compute H(q\_i⊕C(r’)⊙s).

这意味着从正确性角度看 我们所得到的协议与之前的协议功能完全相同

So, it means that for the correctness, we get the same thing as the previous protocol has.

在我们的协议中 Bob可以计算任意哈希值 而Alice只能知道其中一个哈希值

So here, Bob can compute any hash values, and Alice might know one of them.

这就是为什么我们称我们的协议是∞选1-OT扩展的原因

So, that is the reason we call our protocol as 1-out-of-∞ OT extension.

从安全性看 我们要计算C的最优输出比特长度 让此长度等于λ 从而得到安全性

So, for the security, we need to compute the optimal length for the output C, this is λ, to get the security.

从前面两页幻灯片中 我们可以得到幻灯片上的这一行结论

So, from the two previous slides, we have this one.

And now, we use a C as the kind of pseudo-random functions.

So, we just need property of C(r’)⊕C(r\_i) has Hamming weight is negeligible.

So, that is what we need.

And we compute C as the pseudo-random functions with the output length 3.5k.

So, this is what we need for the security of our protocol.

And it means that we need to extend the width of base OT matrices to 3.5k.

So, we need to use more base OT to do it.

So, we use a very small trick, to use pseudo-random generators, to extend the height of OT matrices to 3.5k.

And then, we use a matrix transpose to get a width of base OT matrices as 3.5k.

So, at this point, our protocol achieves the security.

So, here is the whole picture of our protocol.

And again, if r\_i≠r’, F\_(s,q\_i)(r’) looks random now.

So, it means here F\_(s,q\_i)(r)=H(q\_i⊕C(r)⊙s).

And this is exactly oblivious pseudo-random function of F\_(s,q\_i)(r).

So, for each row of OT matrix, it gives us one OPRF instance.

So, that is why we call our protocol as batched OPRF.

And we use different key q\_i for each row, but we have the same s.

So, we get the batched related-key OPRF.

So, that is the reason we call our protocol as batched related-key OPRF.

So now, it is easy to apply OPRF on PSI.

Bob sends F\_(s,q\_i)(r’) to Alice.

Alice simply compares F\_(s,q\_i)(r\_i) and F\_(s,q\_i)(r’), and outputs whether it is equal.

And if r\_i≠r’, F\_(s,q\_i)(r’) looks random to Alice.

So, it means she cannot guess anything about r’.

So, this is exactly private equality test.

So, as the result, our protocol does not depend on the input length |r’| as well as r\_i.

And we have 3 times faster than the current PSI protocol.

So, this is the comparisons of our results on semi-honest PSI.

So, PSI based on the circuit, this is a very general problem.

And so they have the high running time.

So, if we do PSI using public keys, they have the best communications, but they still have high running time.

So, doing PSI based on OT is a good protocol now. They have best running time and good communications.

So, we compare with two recent protocols. We have 3 times faster than the current PSI.

X轴用10为底的对数进行了放缩

So here, the scale is log 10.

我们的协议进一步向朴素哈希算法的结果迈进

And we all move the gap between our protocol to the naive one.

感谢大家的聆听

And thank you for listens.

感谢主讲人的演讲

OK, thank you.

我们还有一些提问的时间

So, we have time for some questions.

你好 有个地方我没有想明白

Hi, so I was just wondering.

你的协议用盲签名等公钥密码学方案构建了OPRF

So, if you did the Oblivious PRF of just a public key base, like a blind unique signature scheme or something like that,

这个过程需要一轮交互 但你提到公钥密码学方案的执行效率很低

so it's one round of communication, and but you're saying it's way slower than this one.

你能再描述一遍问题吗？

Could you speak it again please?

应用OPRF实现PSI的最直接方法是

So, the straightforward way to do this is if you take an OPRF that

Alice要求Bob对双方的字符串签名

Alice asks Bob deploying signature on her string,

Bob对字符串签名 并将结果返回给Alice

and then Bob give me the signature on his string,

Alice检查签名结果是否相等 只要签名结果是唯一的 这个协议就是正确的

and Alice checks if they're equal, if it's a unique signature scheme that works great.

当然这里还有很多限制条件 比如不能直接使用RSA 但是…

I have to stress the condition that cannot be RSA, but…

这个方案效率很高 可以实现PSI

So, that's very fast and that works for PSI.

你前面提到 参与方A要对参与方B所有的字符串签名

You just said you've got signatures on all your strings,

是的

Yes.

参与方A同时要对它自己的字符串签名 随后取两组签名的交集

And signatures on all his strings, you just take the intersection of the signatures.

但你前面提到这个过程效率很低 因为…

But you're saying it's gonna be slow, because…

我明白了 感谢你提出的问题

Yeah, I understand, so thank you for questions.

我们的OPRF只需要使用少量的公钥密码学方案 但需要使用大量的对称密码学方案

So, because we use OPRF, so it means our construction using small public keys and a lot of use symmetric keys.

这就是我们的协议要比公钥密码学方案的协议高效的原因

So, that is reason that our protocol is faster than using a public keys.

因为你只需要执行128次公钥密码学操作吗？

Because you're only doing like 128 public key operations.

是的 我们只需要执行128次 你可以看这页幻灯片

Yeah, we just make 128, yeah you can look at.

这里的技巧在于… 稍等一下

So, here the matrix… Just wait.

我们在OPRF协议中使用了OT扩展协议

So, we use OT extension inside the OPRF protocols.

这意味着我们只需要执行128次公钥密码学操作

So, it means that we just use some public keys 128 some kind of.

明白了

Right.

是的 我们还使用了对称密码学

Yes, and we use some symmetric keys.

很显然 对称密码学的执行效率比公钥密码学高得多 所以…

And of course, symmetric key is cheaper than public key, so…

明白了

Right.

好的 谢谢

Yeah, thanks.

所以这个方案才能这样高效地完成计算 明白了

So, the advantage of the simpler scheme is just speed. That is right, yes, OK.

是的 谢谢

Yeah, thank you.

听众还有什么其它问题吗？

OK, any other questions from the audiences?

没有其它问题了 我们再次感谢主讲人

That is not the case, then thank you for your talk.