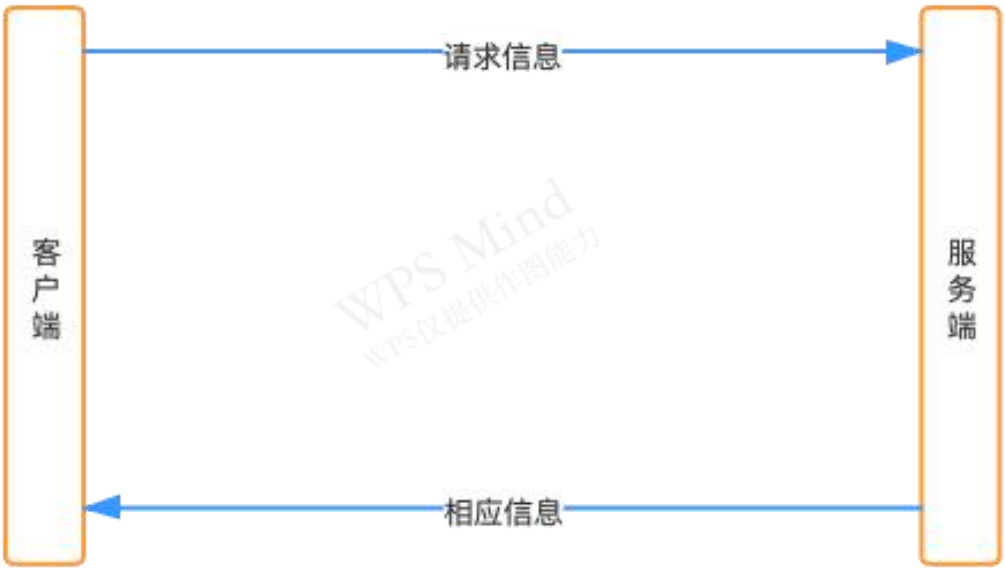
HTTPS 原理

1.未加密状态：



未加密图解

缺点：如果以明文进行传输,那么明文就会跑在网络上,在这个过程中如果有黑客进来,传输过程中的数据将会完全暴露给黑客。

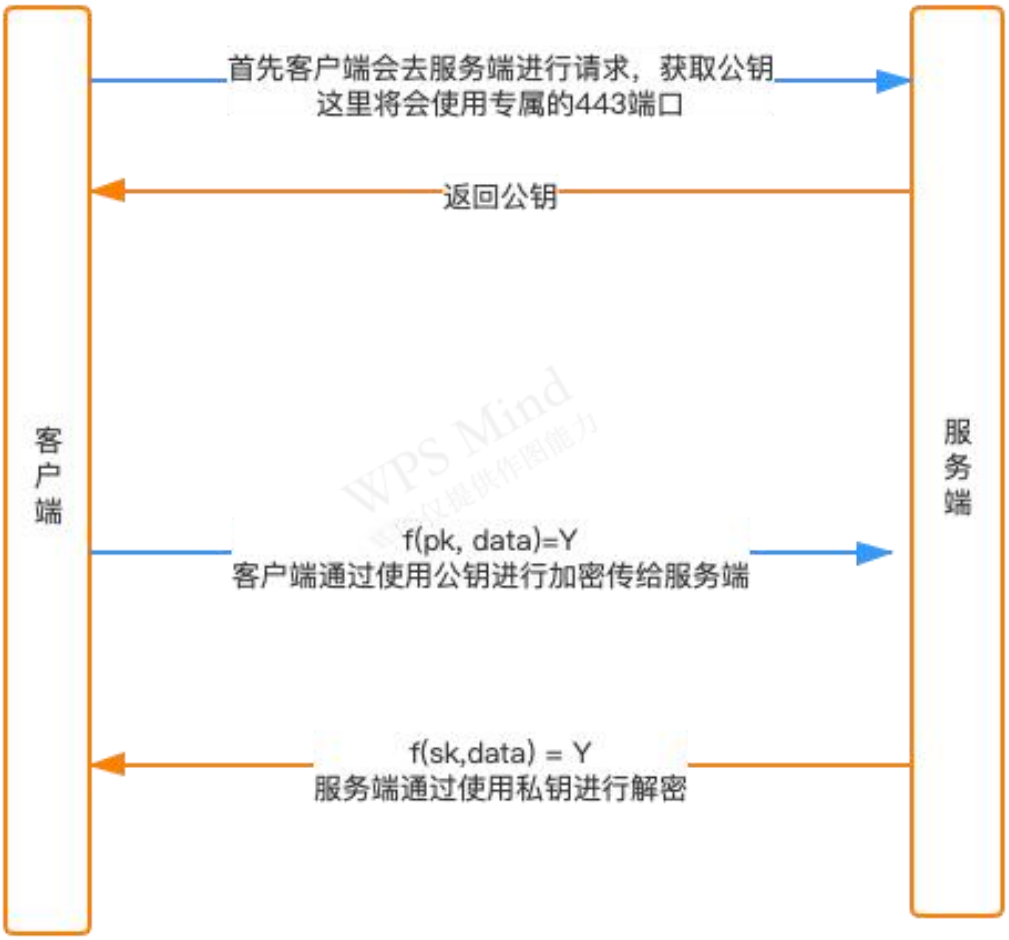
2.使用对称加密算法

对称加密图解

1)f1(key,data)=x 经过加密算法得到加密后的数据得到 x。
2)f2(key,x) = data 经过对应的解密算法进行解密,获取原数据。
3)服务端和客户端每次都是进行先加密,再传输,再解密,处理后再加密发送的过程。
4)由于服务端不可能知道有多少个客户端,也不可能把全部 key 都保存到服务端,所以 key 只能有一个。
5)如果黑客在中间获取到了 key 也可以进行加密解密,那么加密操作将是无效的。

缺点：key 只有一个

3.使用非对称加密

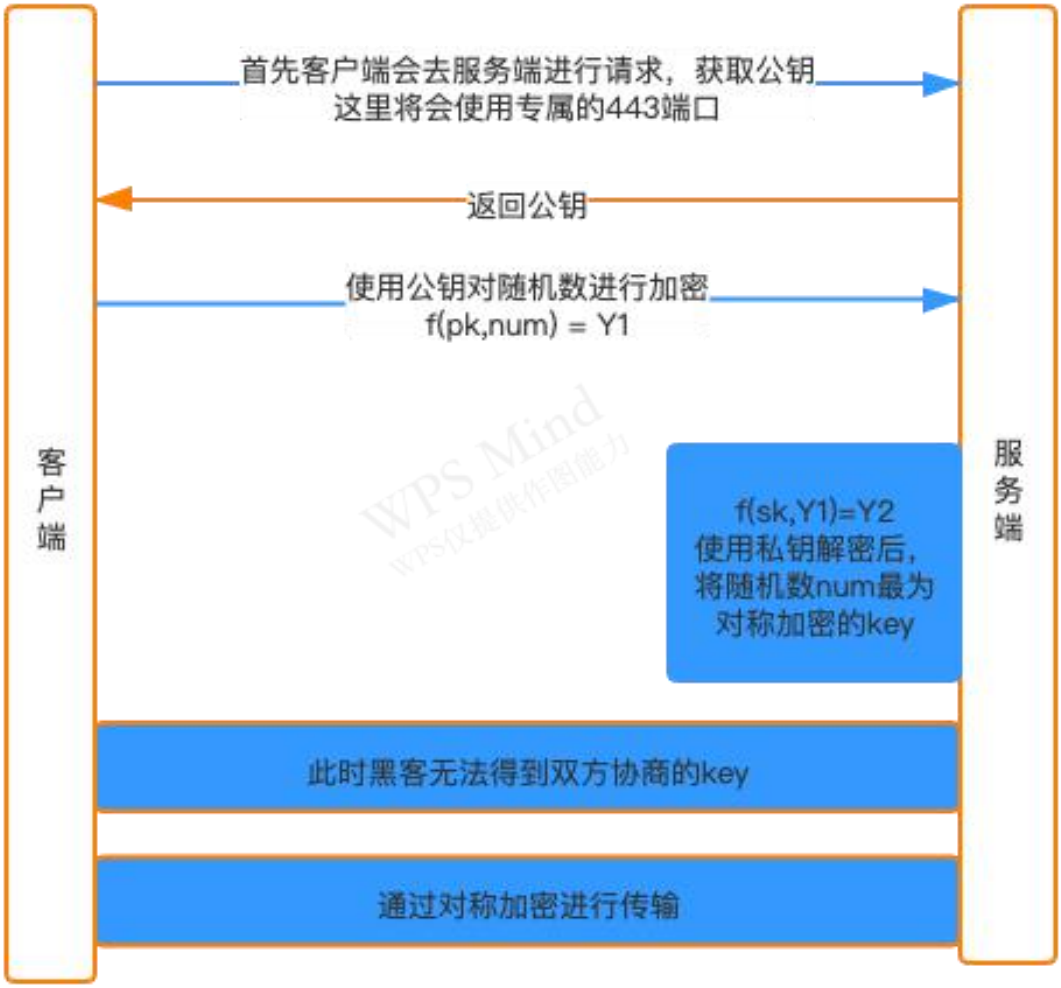


非对称加密图解

处理步骤:
1)首先客户端会去客户端进行请求获取公钥,这里将会使用专属的 443 端口。
2)f(pk,data) = Y 客户端通过使用公钥进行加密传给服务端。
3)f(sk,data) = Y 服务端通过使用私钥进行解密进行处理。
4)同样如果服务端使用私钥进行加密,发送给客户端后客户端也可以通过公钥进行解密。
5)因为私钥只有服务端有其他客户端都没有,所以只有服务端可以进行解密。

但是这个过程有个致命缺点：服务端如何向客户端发送数据?
因为黑客也是可以拿到公钥去解密服务端发来的数据的,所以非对称加密也不可取。

4.对称加密与非对称加密混合



对称加密与非对称加密混合

处理思路：先利用非对称加密的方式来达成服务端与客户端的协商,协商出一个临时制定的 key,之后再使用对称加密进行数据的交互,这样就可以保证 key 的唯一性,每个客户端将有不同的 key。

处理步骤：

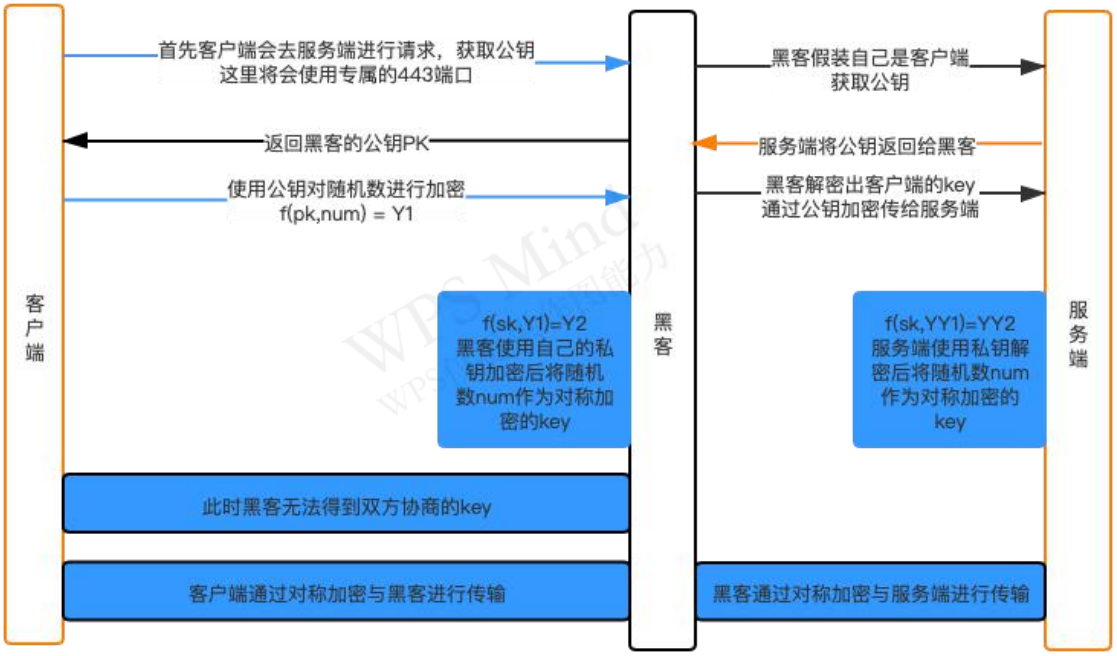
1)第一步还是会去索要公钥,客户端拿到公钥,使用公钥对一个【随机数】进行加密,服务端对拿到的数据进行解密,解密之后的【随机数】就作为服务端与客户端进行对称加密的 key 了。

2)服务端会通知一声客户端 OK,这样就表示协商完成。

3)此时黑客在中间进行截取,但是却并不知道这个协商的 key 是什么,所以也无法进行破解。

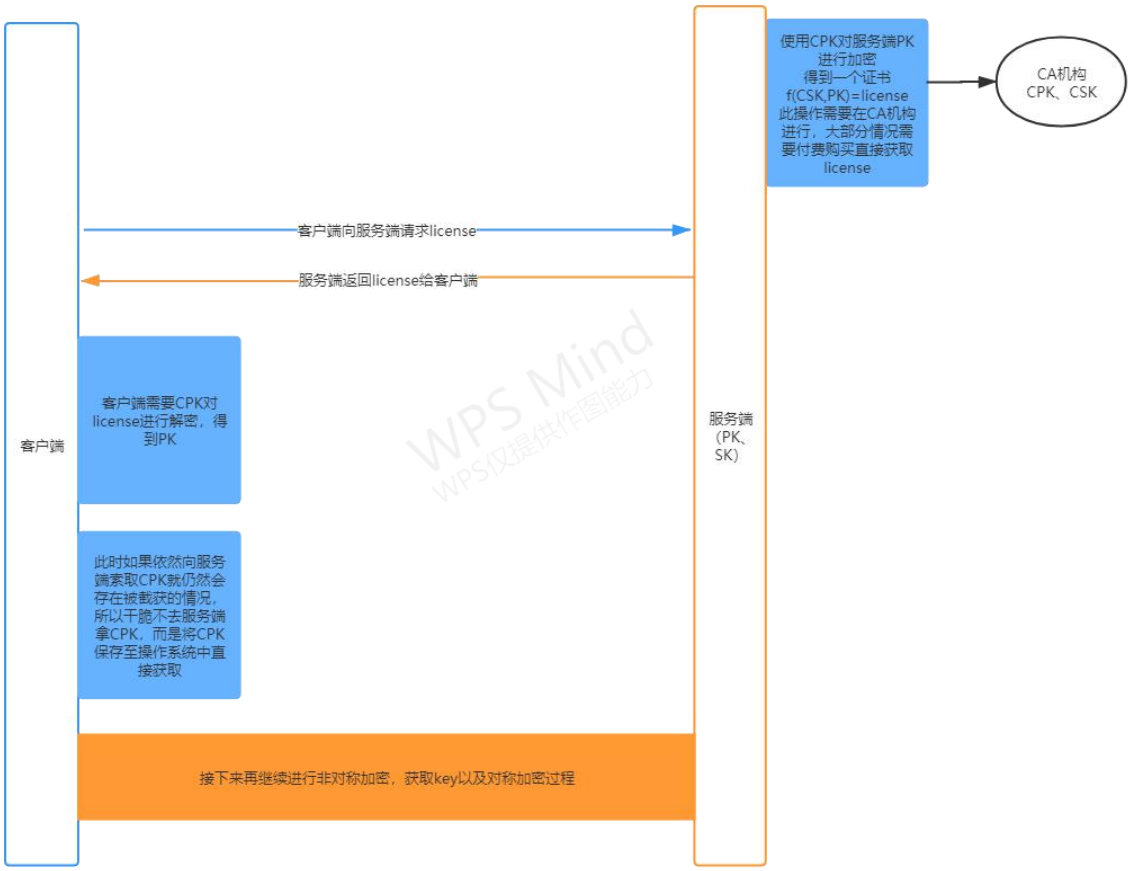
5.中间人攻击:

黑客还是有办法破解的,当客户端去请求公钥的时候,黑客进行拦截,黑客本身也有公钥和私钥,黑客将自身的公钥传递给客户端,黑客假装自己就是客户端去像服务端获取公钥自己留着,客户端通过黑客的公钥进行加密发送给了黑客黑客使用私钥进行解密,那么就可以拿到协商的 key 了,黑客通过使用客户端的 key,再使用自己的公钥进行加密向服务端发送请求,服务端接收到数据进行私钥解密再加密,又会发送给黑客,如此循环,黑客就可以获取到全部的信息了,这就是中间人攻击。



中间人攻击图解

6.引入 CA 证书:



引入 CA 证书图解

1)首先服务端还是有 PK 和 SK ,CA 机构也有 PK 和 SK,这里称为 CPK 和 CSK ,f(CPK,PK)加密之后得到一个证书,我们称为 license,客户端不请求之前的 PK 了,而是直接去请求 license。

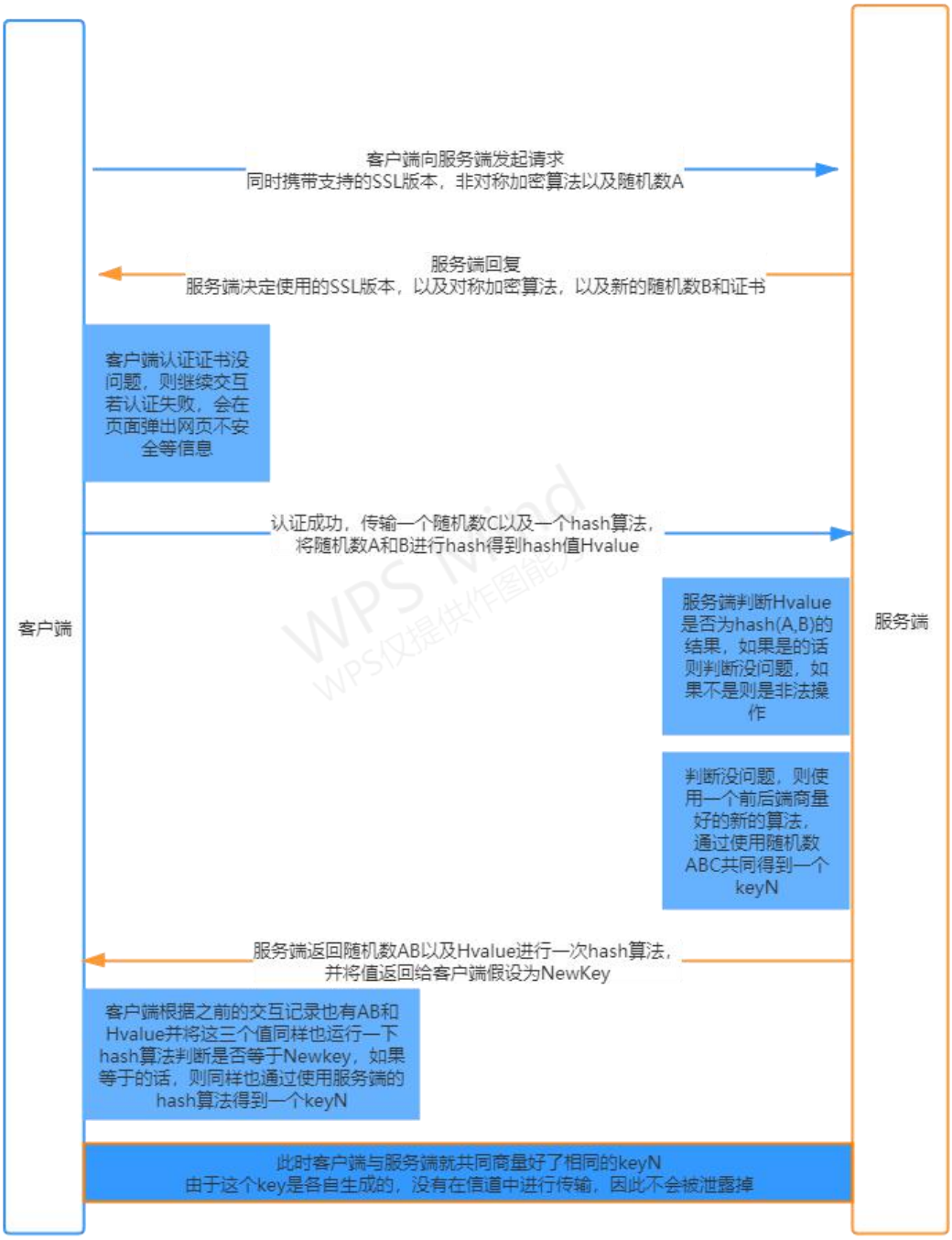
2)那么客户端获取之后如何进行解密呢？去 CA 机构去获取 CPK？但是这样又可能被中间人截获,所以我们干脆不去拿了,直接写死在了操作系统,直接在操作系统端对 license 进行解密就可以拿到 PK 了。

3)获取之后就开始重复协商之后进行交互的过程。

4)此时如果中间人去获取证书,那么这个黑客去解密再加密之后传给客户端,客户端就会发现这个证书是不安全的（很多证书需要付费),网页就会弹出证书不安全,这样用户就不会进行访问了。

5)如果黑客在证书传递之后进行介入,那么此时客户端是同过 PK 进行加密的,而黑客没有对应的 SK 所以也是对密文无能为力的。

7.协商对称加密 key 详细流程



协商对称加密 key 流程介绍

总结：HTTPS 使用了对称加密+非对称加密+hash 算法+CA 认证 4 种技术。

思考：第二个阶段使用非对称加密也是可以的,为什么没有使用非对称加密？？？

答: 理论上是可以的,但是因为非对称加密解密计算开销远远大于对称加密,这是最重要的原因,尤其是服务端如果大量消耗性能会使服务质量下降,而且服务端获取每个客户端的公钥也是很大的开销。