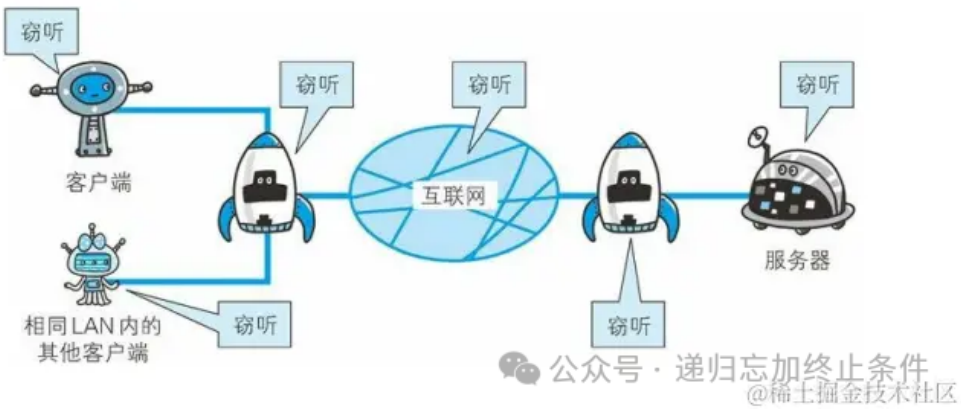
为什么面试官在面试中都爱问 HTTPS？

原创作者：moment 2025年01月15日 12:08* 广东

尽管 `HTTP` 在我们的项目中应用已经很广泛了,然而 `HTTP` 并非只有好的一面,事物皆具有两面性,它也是有不足之处的。

HTTP 的不足之处主要有以下几个方面：

1. 数据传输不加密：HTTP 传输的数据是明文的，任何人都可以在网络中监听并读取传输的数据。这意味着，如果通过 HTTP 传输的是敏感信息（如用户名、密码、银行卡号等），就会容易被窃取。这就会导致数据泄露，影响用户隐私和安全。
2. 数据容易被篡改：HTTP 不提供数据完整性保护，数据在传输过程中可以被中途篡改。恶意攻击者可以通过中间人攻击（Man-in-the-Middle, MITM）修改数据，导致用户接收到被篡改的内容，如篡改的文件、消息等。
3. 缺乏身份验证：HTTP 协议本身无法验证客户端访问的是合法的服务器，可能会遭遇伪造网站或钓鱼网站。攻击者可以通过创建假网站诱导用户输入个人信息或执行恶意操作，造成信息泄露或财产损失。
4. 容易遭受中间人攻击（MITM）：由于 HTTP 协议的数据是明文传输的，攻击者能够通过中间人攻击拦截、读取、修改传输数据。攻击者可以截获会话内容，窃取敏感信息，甚至伪造响应返回给客户端，造成严重的安全隐患。如下图所示：



20250115120338

5. 缺乏数据完整性保护：HTTP 协议本身没有内建的校验机制来验证数据是否在传输过程中被篡改。恶意攻击者可以修改数据，客户端无法判断是否收到被篡改的内容。
6. 浏览器安全警告：许多现代浏览器已经将 HTTP 网站标记为“不安全”，并警告用户。HTTP 网站会影响用户信任，特别是在涉及电子商务、登录、支付等敏感操作时，用户会更加倾向于避免访问 HTTP 网站。
7. 不支持 HTTP/2 特性：HTTP 协议（特别是 HTTP/1.x 版本）效率较低，无法充分利用现代网络的性能优势。比如，它存在队头阻塞（Head-of-Line Blocking）问题，多个请求必须按顺序处理。在大流量的网站或复杂的请求/响应场景下，HTTP 的性能较差，响应速度较慢。
8. 搜索引擎优化（SEO）劣势：搜索引擎（如 Google）更倾向于优先排名 HTTPS 网站，HTTP 网站的排名可能会受到影响。如果一个网站仅使用 HTTP 协议，其搜索引擎排名可能会比使用 HTTPS 的网站低，从而减少网站的访问量。

什么是 HTTPS

为了解决上述存在的问题，就用到了 `HTTPS`，实际上它也并发是应用层的一种新协议，只是 `HTTP` 通信接口部分用 `SSL` 和 `TLS` 协议代替而已。

在正常情况下，`HTTP` 直接和 `TCP` 通信，当使用 `SSL` 时，则演变成先和 `SSL` 通信，再由 `SSL` 和 `TCP` 通信了，换句话说，所谓的 `HTTPS` 实际上就是身披 `SSL` 协议这层外壳的 `HTTP`。

在采用 `SSL` 后，`HTTP` 就拥有了 `HTTPS` 的加密、证书和完整性保护这些功能。

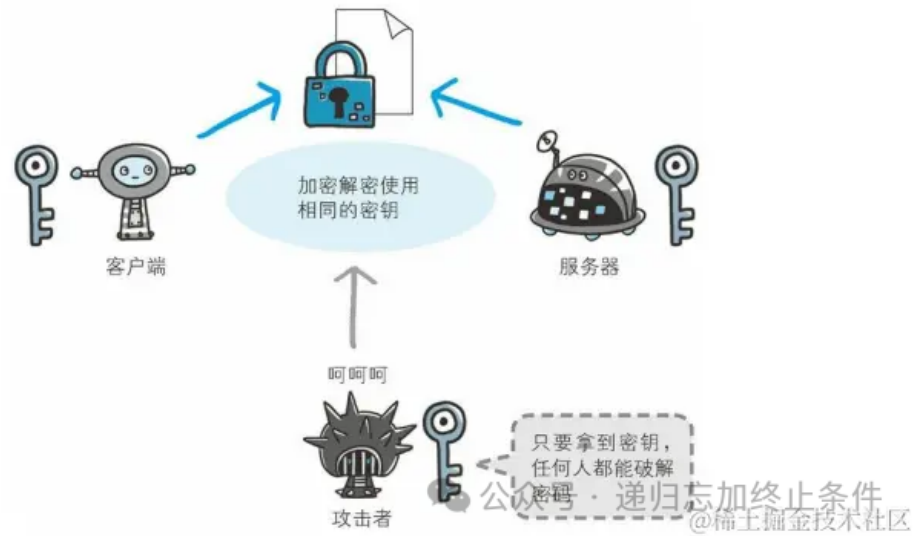
相互交换秘钥的公开密钥加密技术

在对 SSL 进行讲解之前，我们先来了解一下加密方法。SSL 采用一种叫做公开密钥加密的加密处理方式。

在近代的加密方法中，加密算法是公开的，而密钥是保密的，通过这种方式得以保持加密方法的安全性。加密和揭秘都会用到密钥，没有密钥就无法对密码解密，反过来说，任何人只要持有密钥就能解密了。

对称密钥加密(共享密钥加密)

加密和揭秘同用一个密钥的方式称为共享密钥加密，也被叫做对称密钥加密：



20250115120355

以共享密钥方式加密时必须将密钥也发给对方，这是一个挑战，因为在传输密钥本身也需要保证其安全性。如果密钥在传输过程中被截获或篡改，通信的机密性将会被威胁。

在使用共享密钥的通信中，通信双方必须共享同一个密钥，并且双方都必须信任这个密钥的安全性。如果这个密钥在任何一方处被泄露或公开，通信的机密性将无法得到保证。因此，确保双方对共享密钥的安全性保持信任是至关重要的。

我们先来看一个对称加密的例子，假设用户 A 想给用户 B 发送一条加密信息：

1. 用户 A 和用户 B 事先共享一个密钥 `K`。
2. 用户 A 使用密钥 `K` 对消息 `M` 进行加密，生成密文 `C`：`C = E(M, K)`，其中 `E` 是加密算法。
3. 用户 A 将密文 `C` 发送给用户 B。
4. 用户 B 收到密文后，使用相同的密钥 `K` 解密，恢复原始消息 `M`：`M = D(C, K)`，其中 `D` 是解密算法。

对称密钥加密的缺点非常明显

1. 双方需要事先共享密钥，密钥传输过程容易被截获。如果密钥泄露，通信安全将受到严重威胁。

2. 不适合大规模使用：在多方通信中，每对通信方都需要一个独立的密钥。密钥数量增长迅速，难以管理。例如，若有 1000 个用户，每两人之间需要一个密钥，总共需要约 50 万个密钥。
3. 无法实现身份验证：对称加密本身无法验证通信方的身份，容易受到中间人攻击。对称加密本身无法验证通信方的身份，容易受到中间人攻击。

非对称密钥加密(公开密钥加密)

公开密钥加密方式很好地解决了共享密钥加密的困难。它使用一对非对称的密钥,一把叫作私有密钥,另外一把叫作公开密钥。私有密钥不能让其他任何人知道，而公开密钥则可以随意发布，任何人都可以获得。

使用方式: 发送密文的一方使用 `对方的公钥` 对信息进行加密,对方接收到被加密的信息后再使用自己的私钥进行解密。

特点: 信息传输一对多,服务器只需要维持好一个私钥就能和多个客户端进行加密通信。可以实现安全的身份验证、数字签名和密钥交换等功能。

优点:

1. 安全性高: 私钥不会被公开传输,只有私钥的持有者才能解密加密的信息;
2. 方便的密钥交换: 发送方和接收方只需交换公钥,而无需交换密钥;
3. 可以实现数字签名: 私钥持有者可以使用时要对消息进行签名,接收方可以使用公钥验证签名的有效性;

缺点:

1. 计算复杂度高: 与对称密钥加密相比,非对称密钥加密的计算速度慢,处理大量数据时可能会更耗时;
2. 密钥管理复杂: 由于涉及到公钥和私钥的生成、发布和保护,密钥管理可能会更复杂;
3. 通信效率较低：由于加密和解密操作需要使用较长的密钥,导致加密数据的大小增加,从而降低了通信效率;

虽然说安全性高,但也不是没有被盗的可能,因为公钥是公开的,谁都可以获取,如果发送的加密信息是通过私钥加密的话,有公钥的黑客就可以用这个公钥来解密拿到里面的信息。

下面有一个例子，假设用户 A 想发送一条安全消息给用户 B：

1. 用户 A 获取用户 B 的 `公钥`。
2. 用户 A 使用 B 的公钥对消息 进行加密，生成密文： 其中， 是用户 B 的公钥。
3. 用户 A 将密文 发送给用户 B。
4. 用户 B 收到密文后，使用自己的 `私钥` 解密，恢复原始消息： 其中， 是用户 B 的私钥。

非对称加密是一种安全性极高的加密技术，适用于身份验证、密钥交换和数字签名等场景。尽管速度较慢、不适合大数据加密，但它通过与对称加密结合，可以在现代网络通信中高效地提供安全保障。

为什么非对称加密效率低一点

非对称加密的效率较低主要是由于其算法的复杂性和计算成本较高的特点。以下是一些导致非对称加密效率低的主要原因：

1. 密钥长度较长： 非对称加密需要使用一对密钥，包括公钥和私钥。通常情况下，这些密钥的长度要比对称加密中使用的密钥长得多。较长的密钥长度会导致加密和解密的操作都需要更多的计算时间。
2. 计算复杂性： 非对称加密算法（如 RSA 和 Elliptic Curve Cryptography）涉及到大整数运算、模幂运算等复杂的数学运算。这些运算需要更多的计算资源和时间，因此非对称加密的处理速度较慢。

3. 加密速度较慢： 由于非对称加密的加密和解密操作都使用不同的密钥，因此加密和解密速度都较慢。这使得非对称加密不适合处理大量数据，特别是实时通信和大规模数据传输方面。
4. 密钥管理复杂性： 非对称加密需要管理和保护两个密钥：公钥和私钥。这增加了密钥管理的复杂性，包括生成、存储和分发密钥等方面的挑战。
5. 安全性优先： 非对称加密的设计目标之一是提供更高的安全性，因此牺牲了一些性能。密钥的长长度和复杂的数学运算增加了攻击者破解加密的难度，但同时也降低了效率。

非对称加密效率较低主要源于其复杂的数学运算、较长的密钥长度和双密钥管理需求。这些特性决定了非对称加密在性能上无法与对称加密相比，但它通过提供更高的安全性和灵活性，成为密钥交换、身份验证和数字签名等场景的关键技术。通过混合加密和硬件优化，非对称加密的性能瓶颈可以得到有效缓解，从而实现安全与效率的平衡。

混合加密机制

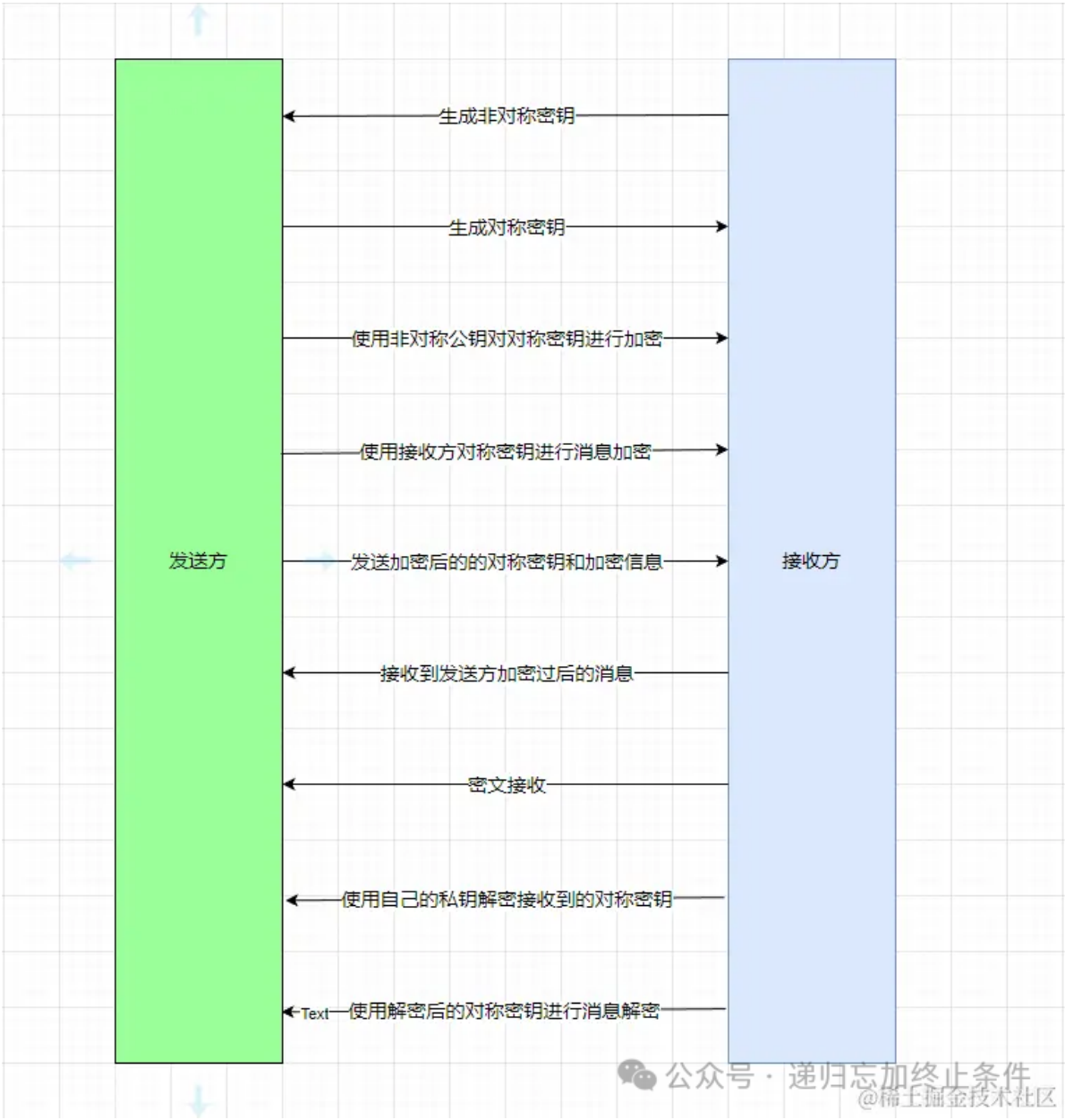
`HTTPS` 采用共享密钥加密和公开密钥加密两者并用的混合加密机制。它采用了对称密钥加密算法的高效性和非对称密钥加密算法的安全性,可以保证安全性的同时提高加密和揭秘的效率。

混合加密机制的操作步骤主要一下几个方面:

1. 密钥交换: 接收方生成一对非对称密钥 (`公钥` 和 `私钥`)，并将公钥发送给发送方;
2. 对称密钥生成: 发送方生成一个随机的对称密钥，用于对消息进行加密;
3. 对称密钥加密: 发送方使用接收方的公钥将对称密钥加密，并将加密后的对称密钥发送给接收方;
4. 消息处理: 发送方使用对称密钥对要发送的消息进行加密，并将加密后的消息发送给接收方;
5. 密文传输: 接收方收到加密后的对称密钥和消息;
6. 对称密钥加密: 接收方使用自己的私钥解密接收到的对称密钥;
7. 消息解密: 接收方使用解密后的对称密钥对接收到的消息进行解密，获得原文明文消息;

在 HTTPS 中，非对称密钥用于安全地交换对称密钥，确保通信双方能在不暴露私密信息的情况下共享加密密钥。之后，对称密钥用于加密和解密实际的数据传输，因为对称加密处理数据速度更快。两者结合确保了数据传输的安全性和效率。

使用文字的方式来表达难免会有些难以理解，接下来我们使用一个流程图来看看混合加密机制的步骤是怎样实现的:



20250115120423

虽然混合加密机制结合了对称加密和非对称加密两者的优势，能够实现双方之间安全的传输。但也不是没有缺点，它的缺点主要有以下几个方面:

1. 数据不完整性: 混合加密主要是为了解决 `HTTP` 中内容可能被窃听的问题。但是它并不能保证数据的完整性，也就是说在传输的时候数据是有可能被第三方篡改的，比如完全替换掉，所以说它并不能校验数据的完整性;
2. 复杂性: 混合加密涉及多种加密算法和密钥管理过程，因此实现和管理起来相对复杂;
3. 密钥交换: 混合加密需要在通信双方之间进行密钥交换，以便建立安全的通信信道，如果密钥交换过程不正确或者被攻击者窃取，那么整个加密系统的安全性将会受到威胁;
4. 性能开销: 混合加密需要同时使用非对称加密和对称加密算法，非对称加密算啊的加密和解密速度较慢，而对称加密算法的加密和解密速度较快。因此，在大规模数据传输时，可能会引入性能开销;
5. 中间人攻击: 混合加密并不能防止中间人攻击，如果攻击者能够劫持或篡改通信信道，并替换公钥或插入恶意代码，那么它们仍然可以窃听、修改或伪装通信内容;

假设用户 A 需要向用户 B 发送加密消息，以下是混合加密的详细过程：

1. 用户 A 生成会话密钥：用户 A 生成一个随机的会话密钥 。例如， 是一个 256 位的对称加密密钥。
2. 用户 A 加密数据**：使用对称加密（如 AES），用户 A 使用 对消息 加密，生成密文：

3. 用户 A 加密会话密钥：使用非对称加密（如 RSA），用户 A 用用户 B 的公钥 加密会话密钥，生成密文：

4. 用户 A 发送数据：用户 A 将加密的会话密钥 和加密的数据 一起发送给用户 B。
5. 用户 B 解密会话密钥：用户 B 使用自己的私钥 解密，恢复会话密钥： 用户 B 使用自己的私钥 解密，恢复会话密钥：

6. 用户 B 解密数据：用户 B 使用会话密钥 解密，恢复出原始消息：

假设用户 B 收到用户 A 通过混合加密机制发送的密文，用户 B 如何通过解密获取明文？以下是完整的解密过程：

1. 解密会话密钥

用户 B 收到加密的会话密钥 和加密的数据密文 。

用户 B 使用自己的**私钥** 对加密的会话密钥 进行解密，恢复出会话密钥：

解密后， 是对称加密所需的密钥。

1. 解密数据密文

用户 B 使用解密得到的会话密钥 对数据密文 进行对称解密：

解密后， 是用户 A 发送的原始明文数据。

混合加密机制结合了对称加密和非对称加密的优点，既保证了数据传输的安全性，又提高了加密处理的效率。这种机制在现代网络通信和数据加密中广泛使用，特别是在 HTTPS 协议、云存储、电子邮件加密和区块链等场景中，成为实现高效安全通信的关键技术。

保证公开密钥正确性的数字证书

目前来看，混合加密机制已经很安全了，但也不是完全没有问题。那就是无法证明公开密钥本身就是货真价实的公开密钥。它有可能在公开密钥传输途中，真正的公开密钥已经被攻击者替换掉了。

为了解决这个问题，通过数字证书认证机构和其他相关机关颁发的公开密钥证书。其中数字证书的基本组成部分主要有以下几个主体:

1. 公钥：证书中包含了公钥，即需要验证的公开密钥;
2. 签名：证书颁发机构使用自己的私钥对证书的内容进行数字签名，以验证证书的完整性和真实性;
3. 有效期：证书包含了开始和结束的有效期，指定了证书的有效期限;
4. 颁发机构信息：证书中包含了颁发机构的身份信息，用于验证颁发机构的可信性;

证书的主体部分包含了公钥持有者的身份信息，如名称、电子邮件地址等。

服务器会将这份由数字证书认证机构办法的公钥证书发送给客户端，以进行公开密钥加密方式通信。接到证书的客户端可使用数字证书认证机构的公开密钥，对那张证书上的数字签字进行验证，一旦验证通过，客户端便可以明确两件事:

1. 认证服务器的公开密钥的真实有效的数字证书认证机构;
2. 服务器的公开密钥是值得信赖的;

数字签名是什么呢，它是一种用于验证数据完整性和身份认证的技术，它的产生过程主要有以下几个步骤:

1. 生成密钥对: 数字签名使用非对称密钥加密算法，首先需要生成密钥对。密钥对包括一个私钥和一个公钥。私钥用于生成签名，而公钥用于验证签名;
2. 签名生成: 使用私钥对数据进行签名，签名生成的过程通常是先对数据进行哈希运算，然后使用私钥对哈希值进行加密，生成签名;
3. 签名附加：将生成的签名与原始数据一起发送或存储;
4. 验证签名：接收方或验证者收到签名和原始数据后，可以执行以下步骤验证签名的有效性
    - 提取公钥: 从签名的来源获取签名者的公钥;
    - 解密签名: 使用签名者的公钥对签名进行解密，得到解密后的哈希值;
    - 哈希计算：对原始数据进行哈希运算，得到哈希值;
    - 比较哈希值：将解密后的哈希值与计算得到的哈希值进行比较。如果两者匹配，说明签名是有效的。如果不匹配，说明签名无效;

通过这个过程，验证者可以确保数据在传输过程中没有被篡改，并且可以确定签名的来源。

数字证书的颁发流程

有了数字签名校验数据的完整性，但是数字签名校验的前提是能拿到发送方的公钥，并且保证这个公钥是可信赖的，所以就需要数字证书。

数字证书的颁发流程通常涉及以下步骤:

1. 密钥生成:
    - 实体(`个人`、`组织`或`服务器`)生成一个密钥对，包括一个公钥和一个私钥;
    - 私钥用于加密和签名，公钥用于解密和验证;
2. 证书请求:
    - 实体向证书办法机构(`Certificate Authority`，`CA`)提交证书请求;
    - 证书请求中包含实体的公钥以及一些身份信息，例如名称、电子邮件地址等;
3. 身份验证:
    - `CA` 对实体的身份进行验证，验证的方式包括人工审核、文件验证、域名验证等;
    - `CA` 确保证书请求的提交者拥有对应的私钥，并具备合法身份;
4. 证书生成:
    - 经过身份验证后，`CA` 使用自己的私钥对证书进行签名，生成数字证书;
    - 数字证书中包含实体的公钥，身份信息以及 `CA` 的签名;
5. 证书颁发:
    - `CA` 将生成的数字证书颁发给实体，通常以电子文件的形式提供;
    - 实体接收到数字证书后，可以将其用于加密通信、数字签名等安全操作;
6. 证书验证:
    - 其他参与者在与实体进行通信时，可以获取实体的数字证书;
    - 参与者使用证书颁发机构的公钥验证证书的签名，确保证书的完整性和真实性;

为什么说数字证书就能对通信方的身份进行验证呢?

数字证书能够对通信方身份进行验证，是因为数字证书采用了公钥加密和数字签名的技术，结合了非对称密钥加密算法的特性。

在数字证书中，证书颁发机构使用自己的私钥对证书进行签名，这个数字签名可以被其他参与这使用 `CA` 的公钥进行验证，通过验证数字签名，可以确保证书的完整性和真实性。

以下几个步骤是数字证书验证通信方身份的过程:

1. 获取证书: 通信方在通信开始之前，从对方获取数字证书;;
2. 提取公钥: 通信方从数字证书中提取对方的公钥;
3. 验证签名: 通信方使用证书颁发机构的公钥对证书中签名进行解密，得到签名的哈希值;
4. 哈希计算: 通信方对原始证书内容进行哈希计算，生成一个哈希值;
5. 比较哈希值: 通信方将解密得到的哈希值与自己计算的哈希值进行比较，如果两者相同，则证书的签名是有效的，证明证书没有被篡改;

通过以上验证步骤，通信方可以确保证书的完整性，并且确定证书的来源是可信的。这样通信方可以信任证书中关联的公钥，并使用公钥进行加密、身份认证或数字签名的验证。

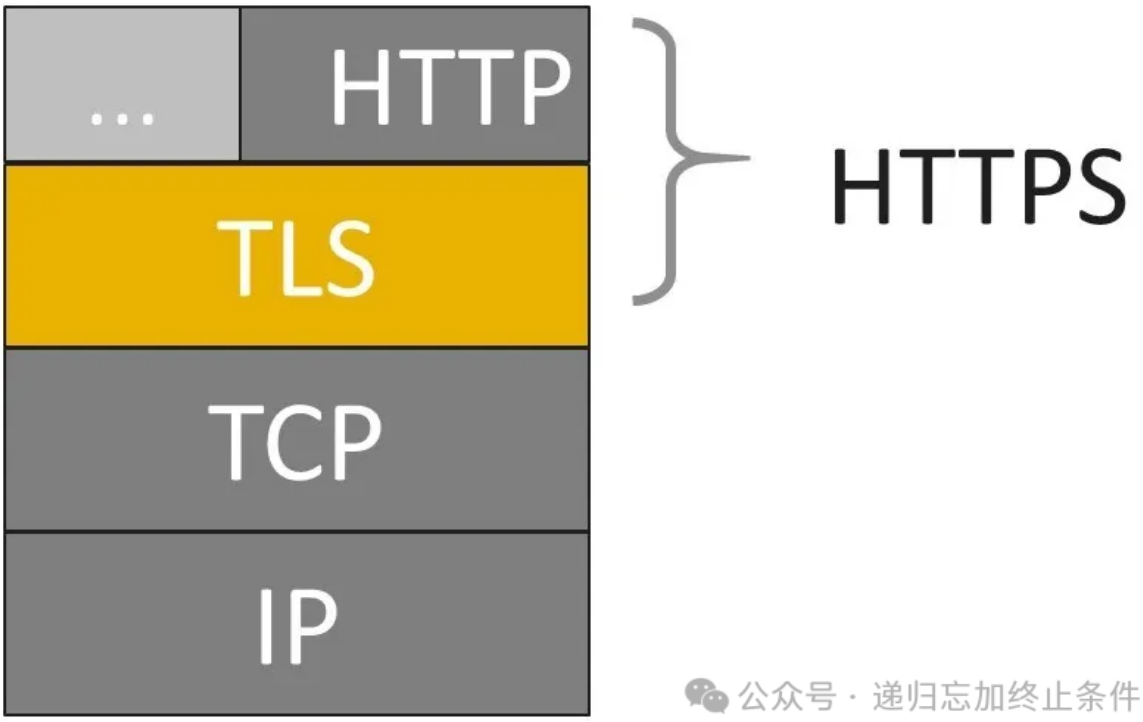
总的来说，数字证书通过使用证书颁发机构的私钥对证书进行签名，提供了一种可信任的方式来验证证书的完整性和真实性。通过验证证书，通信方可以建立对对方身份的信任，并使用其公钥进行安全的通信操作。

SSL/TLS 是如何工作的

HTTPS 是 HTTP 协议的一种安全形式。它围绕 HTTP、传输层安全性 （TLS） 包装了一个加密层。

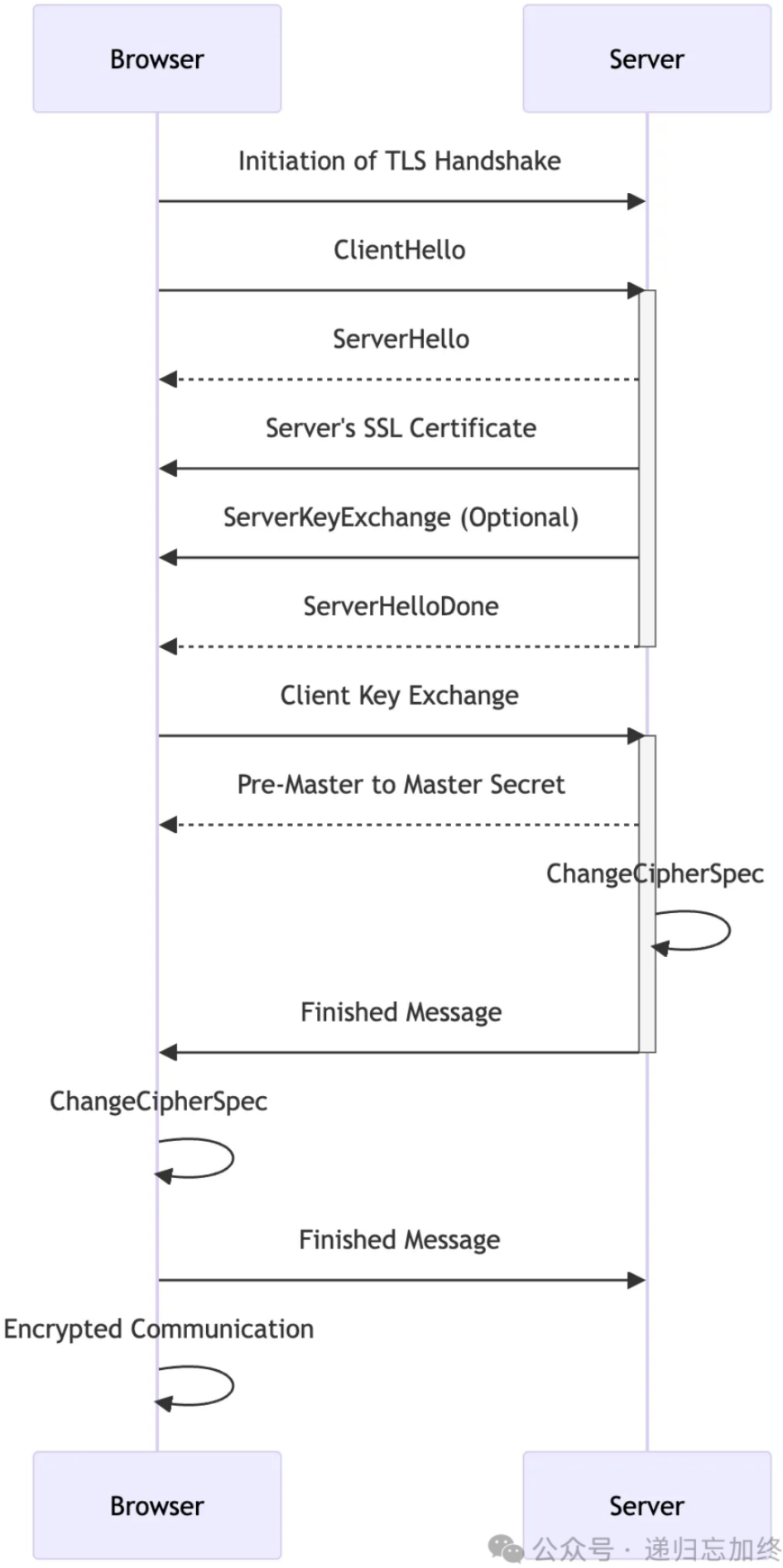
HTTP 只是一种协议，但当与 TLS 配对时，它会被加密。

TLS 和 SSL 是面向 Socket 的协议，因此加密发送方和接收方之间的套接字或传输通道，但不加密数据。这是使这两个协议独立于应用层的主要原因。



20250114143012

接下来我们来看看 TLS 是如何工作的。先上图:



20250114143148

我们将对图中的每一个步骤做详细的解释：

1. 握手启动 (Initiation of TLS Handshake)：浏览器（客户端）发起 TLS 握手请求，与服务器建立安全通信。
2. 客户端问候 (Client Hello)：客户端发送 ClientHello 消息，包含以下内容：
    1. 支持的 TLS 协议版本（如 TLS 1.2、TLS 1.3）。
    2. 支持的加密算法（如 RSA、ECDHE、AES）。
    3. 随机数（用于密钥协商）。
    4. 会话 ID（如果是恢复连接时用）。
3. 服务器问候 (Server Hello)：服务器响应 ServerHello 消息，内容包括：
    1. 确认使用的 TLS 协议版本。
    2. 选择的加密算法。
    3. 服务器生成的随机数。
    4. 会话 ID。
4. 服务器证书（Server Certificate）：服务器发送其 SSL/TLS 证书（由 CA 签发），包含：
    1. 服务器的公开密钥。
    2. 服务器的身份信息（如域名）。
    3. 证书的有效期。
5. 服务器密钥交换 (Server Key Exchange，可选)：在某些情况下（如使用 Diffie-Hellman 密钥交换算法），服务器会发送密钥交换参数。这一步是可选的，具体取决于协商的加密算法。
6. 服务器握手结束通知 (Server Handshake Finished)：服务器发送 ServerHelloDone，表示服务器端的握手阶段完成。
7. 客户端密钥交换 (Client Key Exchange)：客户端生成一个 预主密钥（Pre-Master Secret），并使用服务器的公钥加密后发送给服务器。服务器用私钥解密，得到预主密钥。
8. 生成主密钥（Pre-Master to Master Secret）：客户端和服务器各自使用预主密钥、客户端随机数、服务器随机数，以及协商的加密算法，生成主密钥。
9. 通知切换到加密模式 （Change Cipher Spec）：客户端和服务器分别发送 ChangeCipherSpec 消息，表明后续通信将使用加密模式。
10. 握手完成确认 (Handshake Finished)：客户端和服务器分别发送握手完成确认消息，确认握手过程完成。
11. 加密通信 (Encrypted Communication)：握手完成后，客户端和服务器使用主密钥进行加密通信。

在上面的步骤中，主要有三个核心流程：

1. 身份验证：通过服务器的 SSL/TLS 证书验证其身份。
2. 密钥协商：利用非对称加密生成共享的会话密钥。
3. 加密通信：使用对称加密（如 AES）提高传输效率。

HTTPS 是通过在 HTTP 上加入 TLS（传输层安全协议）实现安全通信的，它提供加密、身份验证和数据完整性保护。TLS 握手是 HTTPS 的核心流程，客户端与服务器通过握手协商加密算法、验证服务器身份，并生成共享的会话密钥。完成握手后，双方使用对称加密对数据进行高效传输，确保通信内容的机密性和完整性。

总结

尽管 HTTPS 提供了显著的安全优势，但由于 性能开销、证书管理成本、特定场景需求 和 历史遗留问题，一些场景下仍然使用 HTTP。不过，随着免费证书的普及、TLS 1.3 的性能提升以及对安全性的重视，使用 HTTPS 已成为现代互联网的趋势，并被搜索引擎（如 Google）优先推荐。

`HTTPS` 的本质就是在 `HTTP` 的基础上添加了安全层,主要是通过他来加密和验证机制来保护通信数据的安全性和隐私性。它提供了保密性、完整性和身份验证的重要机制,使得数据在传输过程中得到了有效的保护,防止数据被窃听、篡改和伪装。