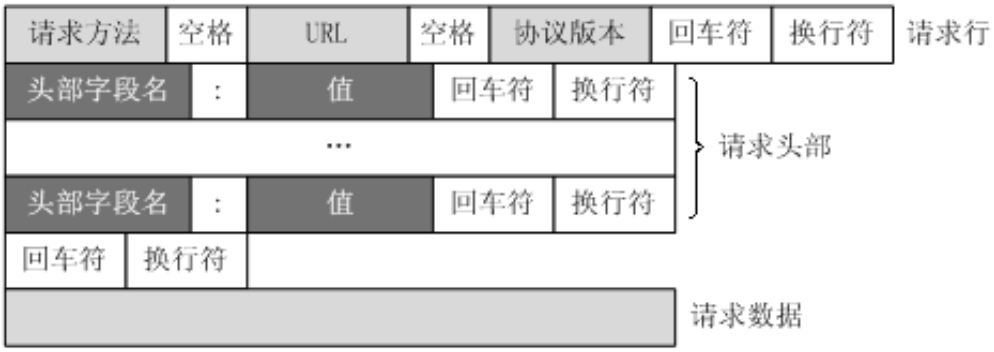
HTTP请求报文

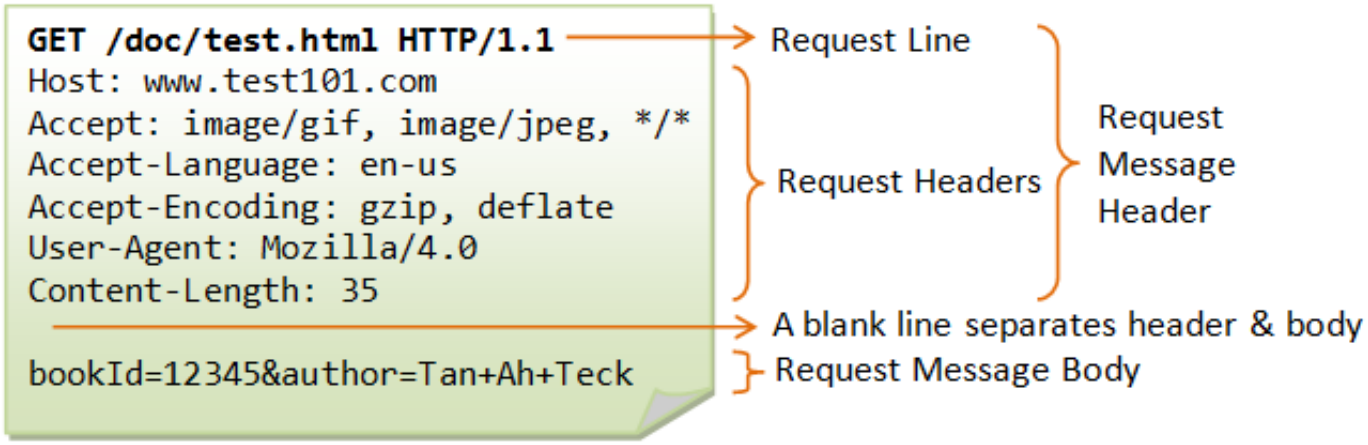
- HTTP 请求报文由请求行（request line）、请求头部（request header）、空行、请求数据四个部分组成。



1. 请求行由请求方法字段、URL 字段、HTTP 协议版本字段构成，用空格隔开。 *GET /index.html HTTP/1.1*
2. 请求头部内容中的典型头部字段名
    - User-Agent： 产生请求的浏览器类型
    - Accept： 客户端能够识别的文件类型列表
    - Host： 请求的主机号

3. HTTP 请求方法字段： GET、HEAD、POST用于 HTTP/1.0

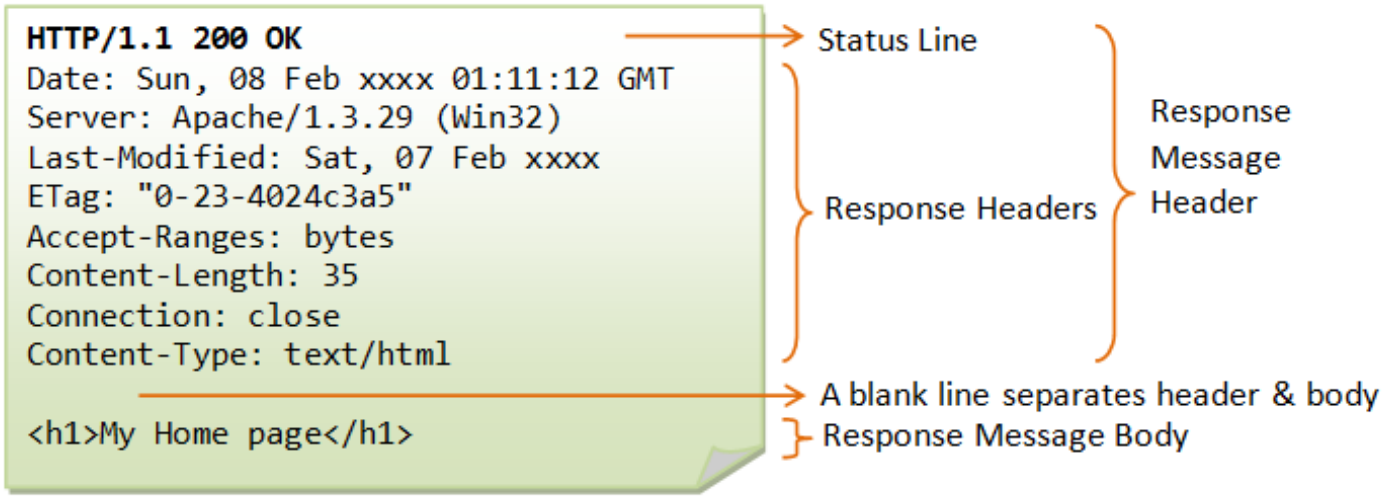
GET	**请求指定的页面信息，并返回实体主体，绝大多数场合使用此方法，用于获取资源**
HEAD	获取报文首部，和 GET 方法类似，但是不返回报文实体主体部分。主要用于确认 URL 的有效性以及资源更新的日期时间等
POST	主要用于传输数据，数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改
PUT	上传文件，自身无验证机制故存在安全性问题，一般不使用
DELETE	与PUT相反，用于删除文件，同样无验证机制
OPTIONS	查询指定的 URL 能够支持的方法。会返回 `Allow: GET, POST, HEAD, OPTIONS` 这样的内容。
CONNECT	建立管道
TRACE	追踪路径



HTTP状态码

状态码	类别	含义
1**	Informational（信息性状态码）	接收的请求正在处理
2**	Success（成功状态码）	请求正常处理完毕
3**	Redirection（重定向状态码）	需要进行附加操作完成请求
4**	Client Error（客户端错误状态码）	服务器无法处理请求
5**	Server Error（服务器错误状态码）	服务器处理请求出错

HTTP响应码



GET与POST的区别

1. GET提交，请求的数据会附在URL之后（就是把数据放置在HTTP协议头＜request-line＞中）， 以?分割URL和传输数据，多个参数用&连接;例如：login.action?name=hyddd& password=idontknow&verify=%E4%BD%A0 %E5%A5%BD。如果数据是英文字母/数字，原样发送，如果是空格，转换为+，如果是中文/其他字符，则直接把字符串用BASE64加密，得出如：%E4%BD%A0%E5%A5%BD，其中%XX中的XX为该符号以16进制表示的ASCII。

2. POST提交：把提交的数据放置在是HTTP包的包体＜request-body＞中。上文示例中红色字体标明的就是实际的传输数据

3. 因此，GET提交的数据会在地址栏中显示出来，而POST提交，地址栏不会改变。

4. 传输数据的大小，GET会收到URL长度限制的局限，POST理论上不受限，实际上受到不同浏览器的限制。

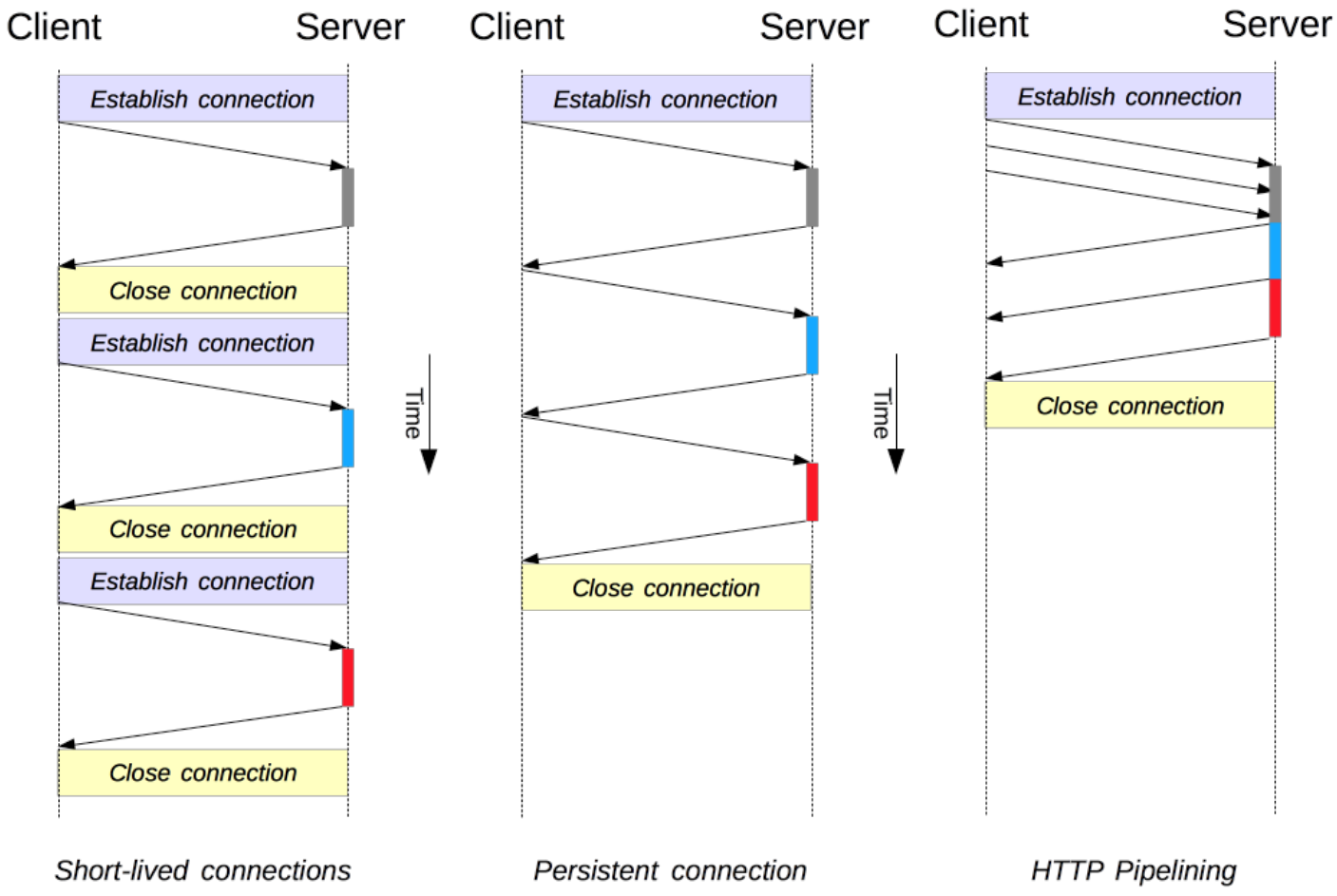
5. GET 安全性较低，因为密码等内容可能会出现在 url 中，或者被浏览器缓存，但是 POST 请求的参数也可能通过抓包工具获取。

6. GET方法是“安全的”，也即是不会改变服务器状态的，而 POST 则不是。

7. 幂等性：指同样的请求多次执行和一次执行结果是否有差别，GET、HEAD、DELETE、PUT 是幂等的而POST不是。

连接管理

- 短连接与长连接：长连接每次TCP连接到可以多次 http 通信，HTTP/1.1 起是默认长连接的，如果要断开连接需要服务器或者客户端提出请求。
- 流水线： 默认情况下HTTP请求是按顺序发出的，只有当前请求收到响应后才会发出下一个请求，然而流水线是在同一个长连接上发出连续的请求而不用等待返回，这样避免连接延迟。

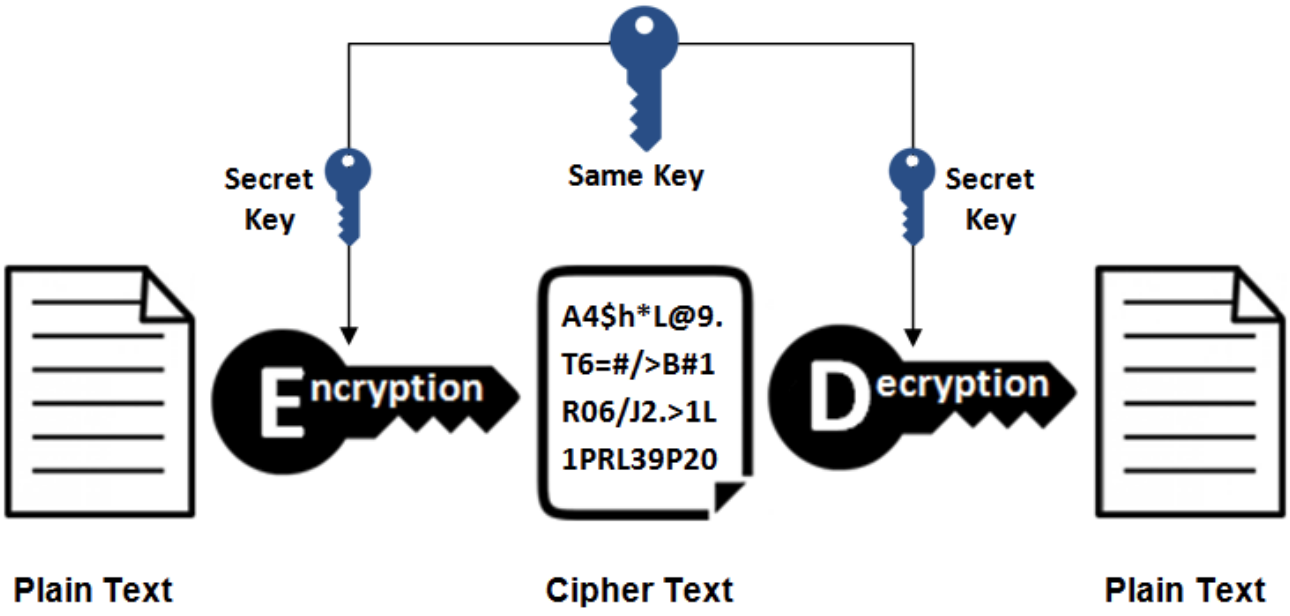


Cookie Session Cache

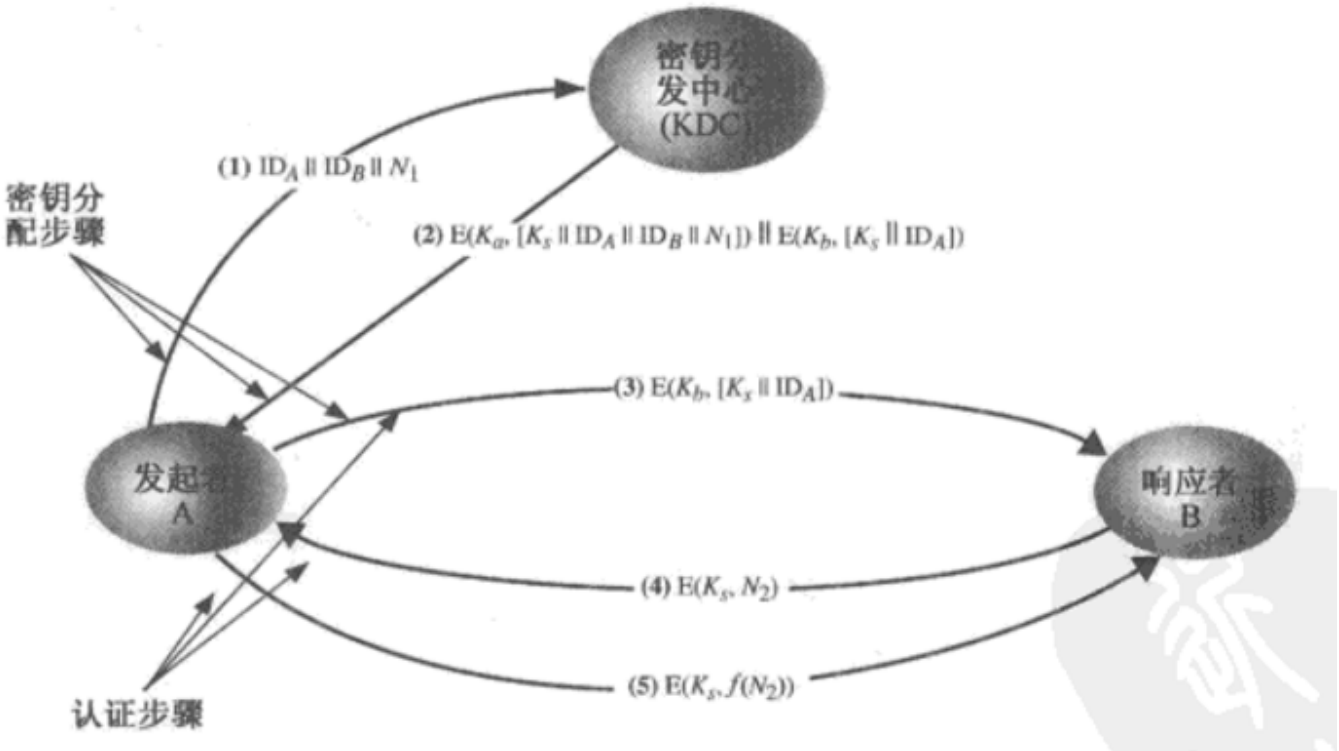
> TODO

HTTPS

- HTTP 有着明文通信导致可能被监听、不验证通信方身份导致遭遇伪装身份、无法证明报文完整性导致报文被篡改的缺点。
- HTTPS 的原理在于先让 HTTP 与 SSL（Secure Sockets Layer）进行通信，随后再由 SSL 与 TCP 通信，也就是说 HTTPS 使用了隧道进行通信。
- ***对称加密*** 的原理在于发送方向接收方发送根据密钥加密的加密文档，再由接收方用同一密钥进行解密。



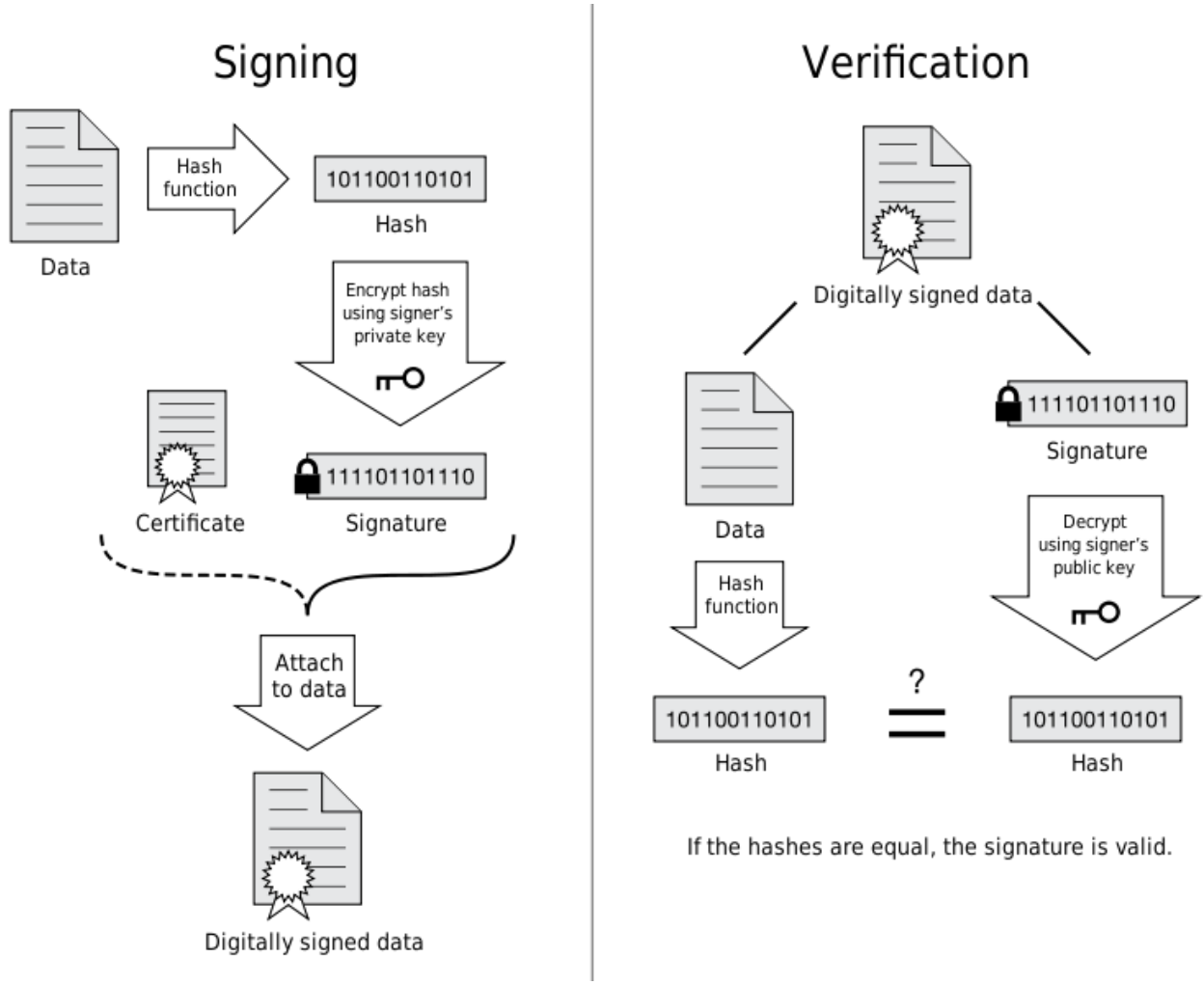
- 这个方法的问题在于如何保证密钥的安全传递，现在常用的方法在于有一个第三方密钥分发中心 KDC，能够与收发者均保持加密连接，利用物理的方式保持与每个终端用户共享一对唯一的主密钥（如U盾），KDC 可以在专用的分发加密连接中将密钥发送给双方。
- 发送者 A 与 KDC 交互的过程。



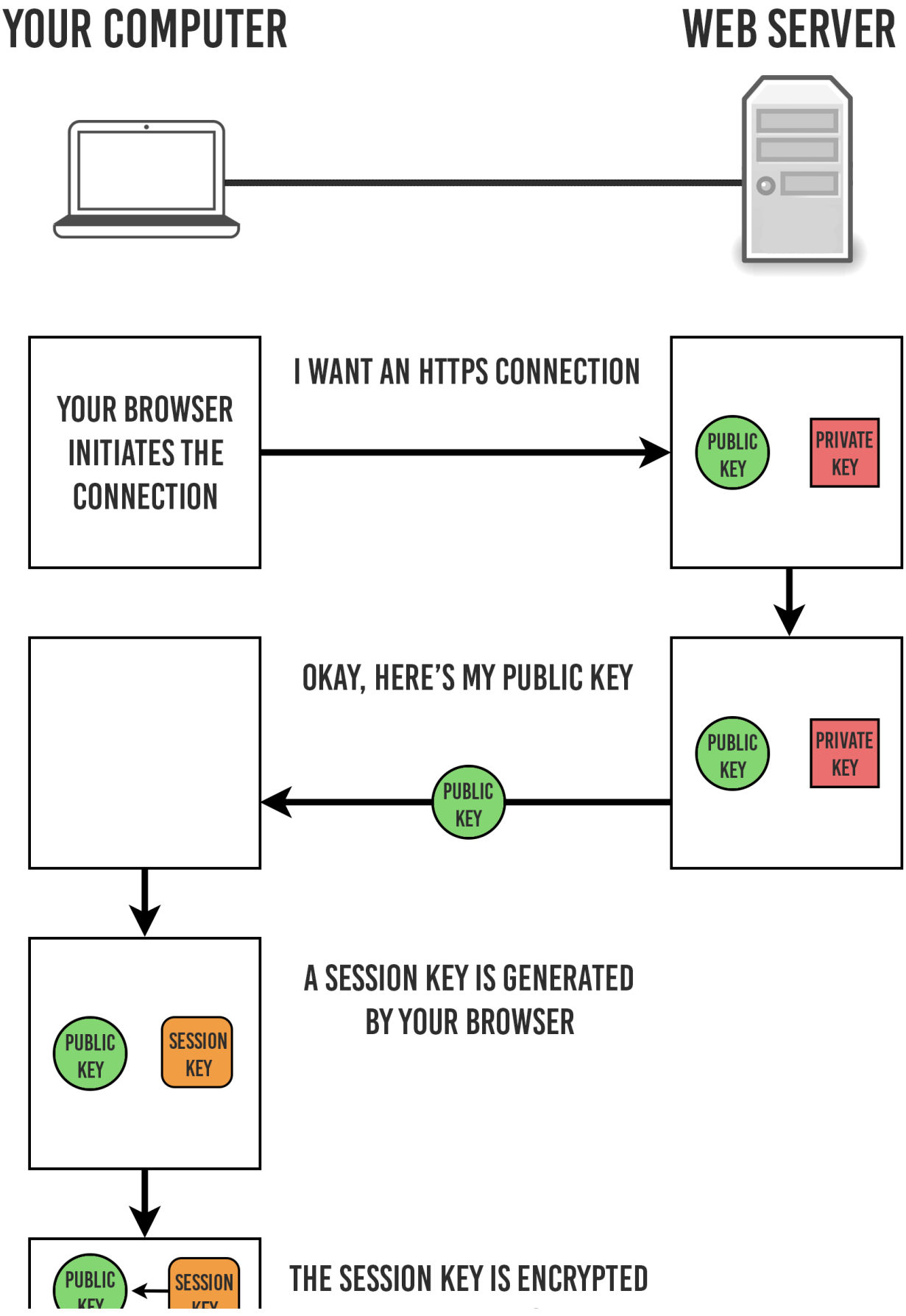
- 优点：算法公开、计算量小、加密速度快、加密效率高。
- 缺点：秘钥的管理和分发非常困难，不够安全。在数据传送前，发送方和接收方必须商定好秘钥，然后双方都必须要保存好秘钥，如果一方的秘钥被泄露，那么加密信息也就不安全了。另外，每对用户每次使用对称加密算法时，都需要使用其他人不知道的唯一秘钥，这会使得收、发双方所拥有的钥匙数量巨

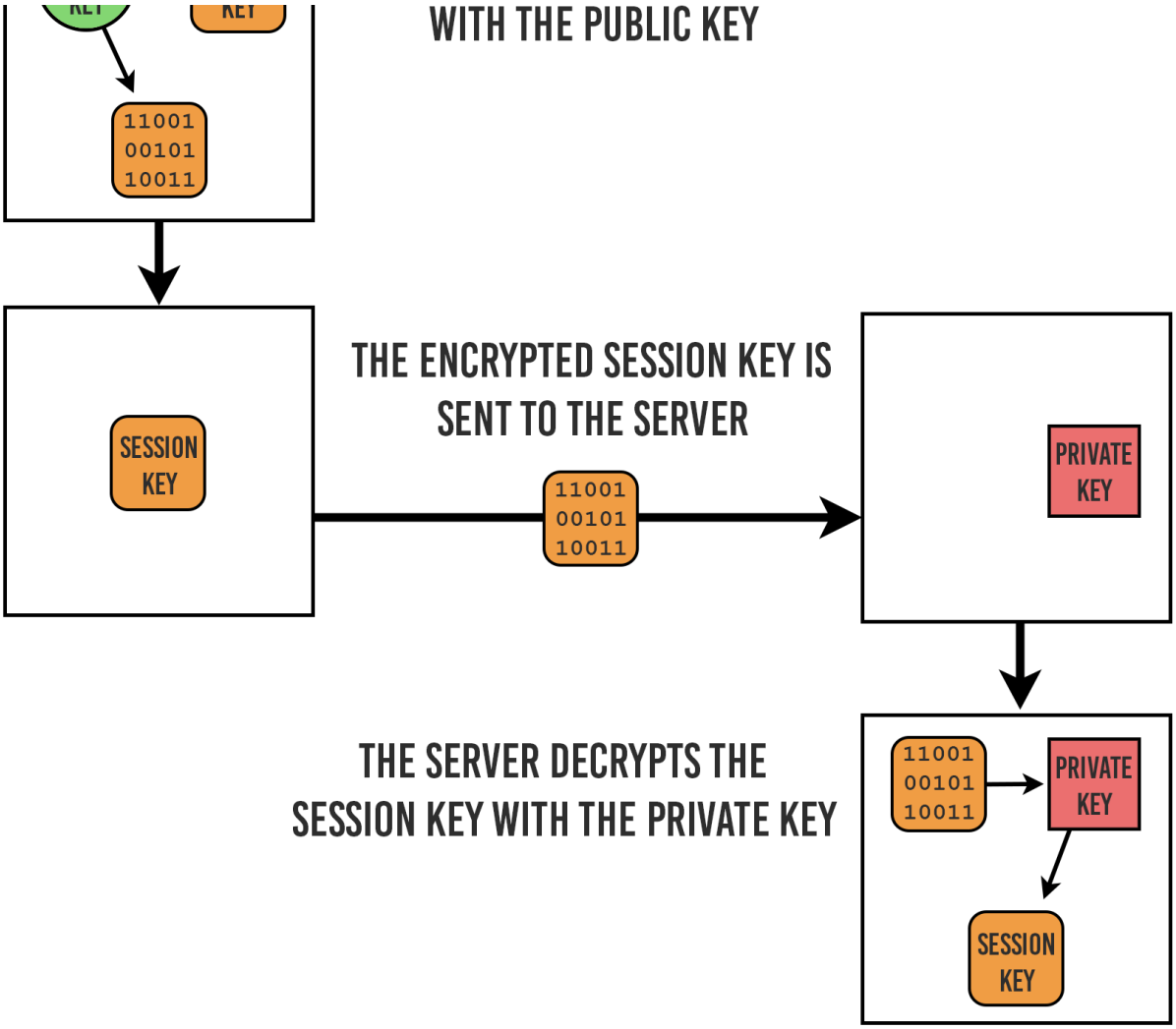
大，密钥管理成为双方的负担。

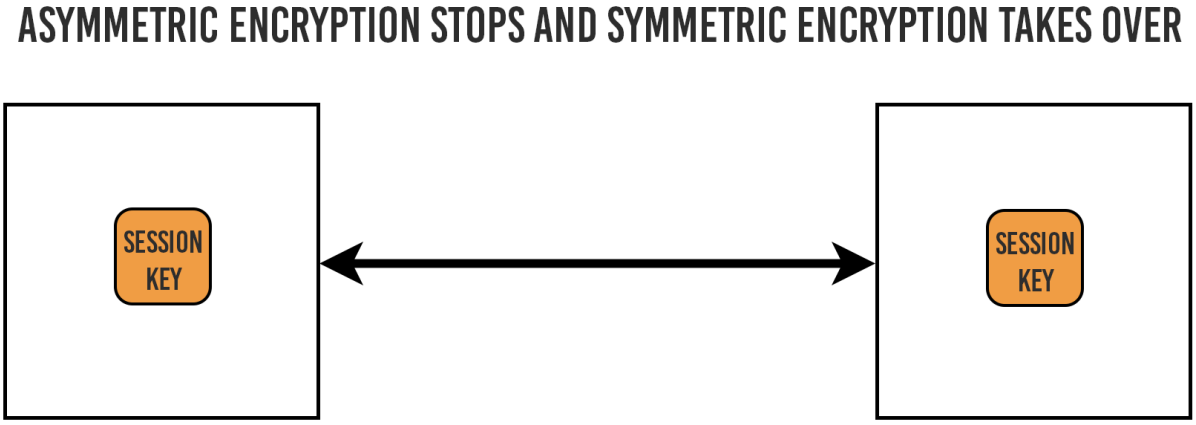
- ***非对称加密*** 又称公开密钥加密，与前者最大的不同在于加密与解密所用的不是一把密钥，公开密钥所有人均可获得，而私钥只有自己知道。
- B 的公钥是公开的，当 A 给 B 发送信息的时候需要用 B 的公钥进行加密，等到消息被 B 接收到之后，B 可以用自己的私钥对信息进行解密，
- 优点：安全性更高，公钥是公开的，秘钥是自己保存的，不需要将私钥给别人。
- 缺点：加密和解密花费时间长、速度慢，只适合对少量数据进行加密。
- CA 客户端浏览器证书管理器 受信任的根证书颁发机构列表
- 服务器的运营人员向 CA 提出公开密钥的申请，CA 在判明提出申请者的身份之后，会对已申请的公开密钥做数字签名，然后分配这个已签名的公开密钥，并将该公开密钥放入公开密钥证书后绑定在一起。进行 HTTPS 通信时，服务器会把证书发送给客户端。客户端取得其中的公开密钥之后，先使用数字签名进行验证，如果验证通过，就可以开始通信了。



HOW HTTPS ENCRYPTION WORKS







- HTTPS 采用混合加密的方法
- 先进行非对称加密的方式来传输对称加密的密钥，保证其安全性，再利用对称加密进行通信保证通信效率。