1. 爬虫的架构
2. 新浪的登录流程

登录前会有prelogin来获取timestamp，nonce，pubkey,servertime,rsakv

1. http和https的区别

https在TCP层和应用层中间加了一个SocketSecureLayer或TransportSecureLayer，在TCP三次握手后，再进行SSL握手，首先客户端给服务器发送自己的浏览器版本，支持加密格式，以及一个随机数。然后服务器给客户端发送一个CA（Certification Authority数字证书）证书以及一个随机数。客户端通过这个CA证书来获取服务器的公钥，然后通过这个公钥加密一个对称秘钥（由三个随机数生成）给服务器。然后服务器用自己的私钥来解密私钥。最后整个会话期间全部用对称的秘钥加密。

此密钥有时间限制，过期会被服务器删掉。秘钥前两个随机数是可以被监听的。所以破解秘钥最关键的是第三个随机数

若ssl中断需要重新连接，但如果有sessionID或者sessionticket的话就不需要重新生成秘钥了

1. session 与 cookie的区别

cookie为保存在本地的数据

session为保存在服务器端的记录，基于cookie实现

cookie存储以下信息

Set-Cookie: name = VALUE;

expires = DATE;

path = PATH;

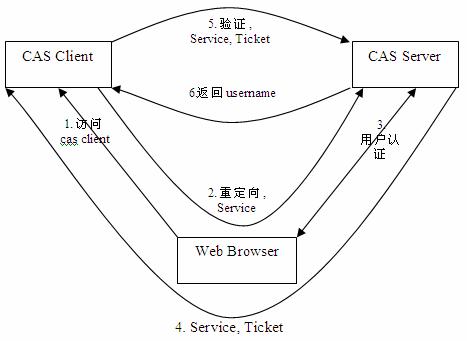
domain = DOMAIN\_NAME;

而session通过在本地保存一个存有sessionid 的cookie，发送给服务器来识别

1. 微博的sso（single sign on）实现了yale-cas

CAS 原理和协议

从结构上看，CAS 包含两个部分： CAS Server 和 CAS Client。CAS Server 需要独立部署，主要负责对用户的认证工作；CAS Client 负责处理对客户端受保护资源的访问请求，需要登录时，重定向到 CAS Server。图1 是 CAS 最基本的协议过程：



CAS 基础协议

CAS Client 与受保护的客户端应用部署在一起，以 Filter 方式保护受保护的资源。对于访问受保护资源的每个 Web 请求，CAS Client 会分析该请求的 Http 请求中是否包含 Service Ticket，如果没有，则说明当前用户尚未登录，于是将请求重定向到指定好的 CAS Server 登录地址，并传递 Service （也就是要访问的目的资源地址），以便登录成功过后转回该地址。用户在第 3 步中输入认证信息，如果登录成功，CAS Server 随机产生一个相当长度、唯一、不可伪造的 Service Ticket，并缓存以待将来验证，之后系统自动重定向到 Service 所在地址，并为客户端浏览器设置一个 Ticket Granted Cookie（TGC），CAS Client 在拿到 Service 和新产生的 Ticket 过后，在第 5，6 步中与 CAS Server 进行身份合适，以确保 Service Ticket 的合法性。

在该协议中，所有与 CAS 的交互均采用 SSL 协议，确保，ST 和 TGC 的安全性。协议工作过程中会有 2 次重定向的过程，但是 CAS Client 与 CAS Server 之间进行 Ticket 验证的过程对于用户是透明的。

另外，CAS 协议中还提供了 Proxy （代理）模式，以适应更加高级、复杂的应用场景，具体介绍可以参考 CAS 官方网站上的相关文档。