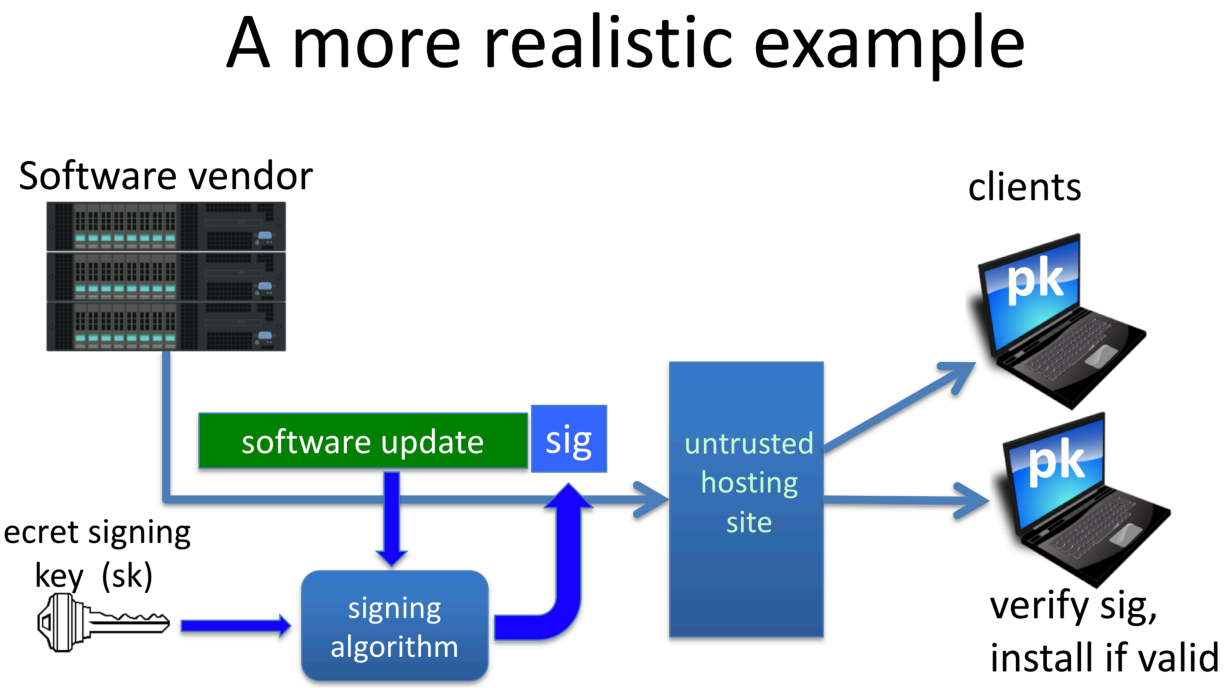
数字签名体系

数字签名体系 : **为了让Clients 认出且识别出, 消息确实是从服务区端发送的.且没有受到完整性的破坏, 因为如果完整性被破坏了, 那么V函数的(message, ...) 就和签名时候的 S(message , ..) 里面的 message 不一样了, 从而验证失败**



基本构成

一个数字签名体系包括三个部分 : Gen() 生成一个pk, sk , sig = S(sk, message) , V(sig, pk, message) 输出"accept" or "reject"

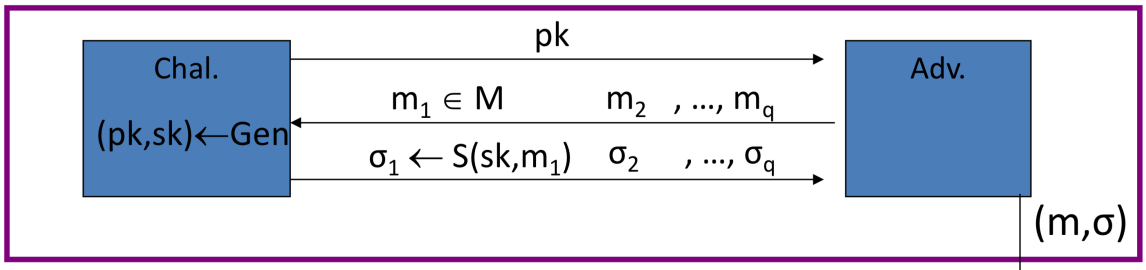
- S()函数是服务器端计算, 用的是sk, 生成一个关于message和sk 的签名 Sig
- V()函数是客户端计算, 用pk 验证签名的合法性, 以及消息是不是由服务器法术
- 理论上 V((sk, message), pk, message) = "accept"

安全方面 existential forgery

攻击者的能力 : 可以获得服务器关于消息 m0, m1, ... m_q 的签名

攻击者目标 : 可以在自己创造一个签名对 (m_j , sig) 使其通过验证 , 其中 m_j 是不存在于 m0, m1, ... m_q中的

For a sig. scheme (Gen,S,V) and adv. A define a game as:



Adv. wins if $V(pk,m,\sigma) = \text{`accept'}$ and $m \notin \{m_1, \dots, m_q\}$

Def: SS=(Gen,S,V) is **secure** if for all "efficient" A:

$Adv_{SIG}[A,SS]$ = Pr[A wins] is "negligible"

CA机构

该机构为第三方机构 , 为了解决如下问题 :

Client端 得到一客户端的pk, 但是怎么能确定, 这个pk就是服务端发送给Client的呢, 这时候就需要CA做验证, 证明确实是服务端提供的pk

数据完整性(data integrity)检测的几个方法

- Collision resisitant hash (抗碰撞的Hash函数) , 需要read-only 的space存放之
- MAC : 一般用于处理 one - to -one 的消息通信机制, 在之前一般需要通信协商一个共享的key
- Signature : 一般用于处理 one - to - many 的消息机制, 即 **S - C** , 需要自己管理一个sk用于数字签名