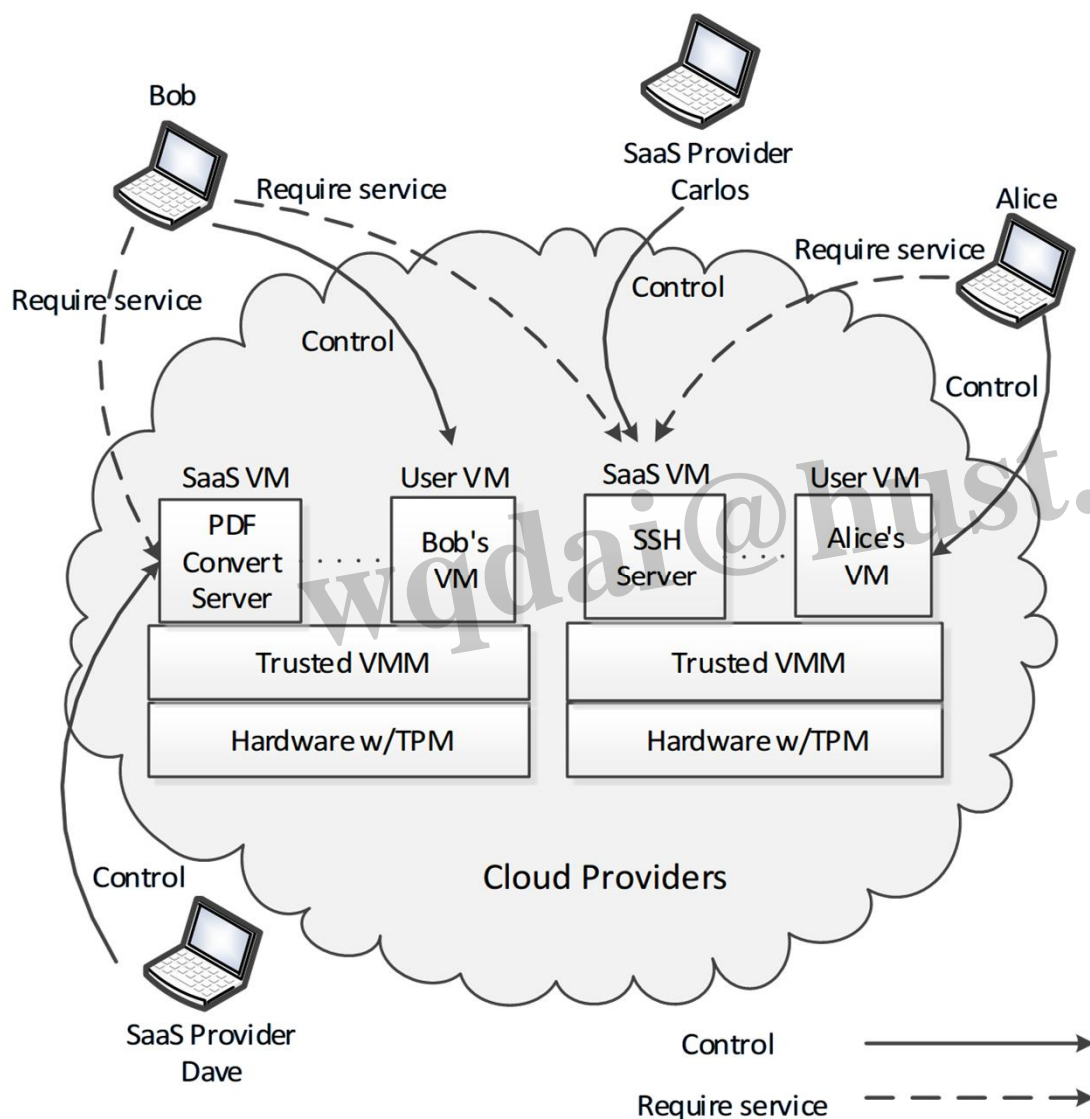


虚拟机安全回滚机制

● 问题:

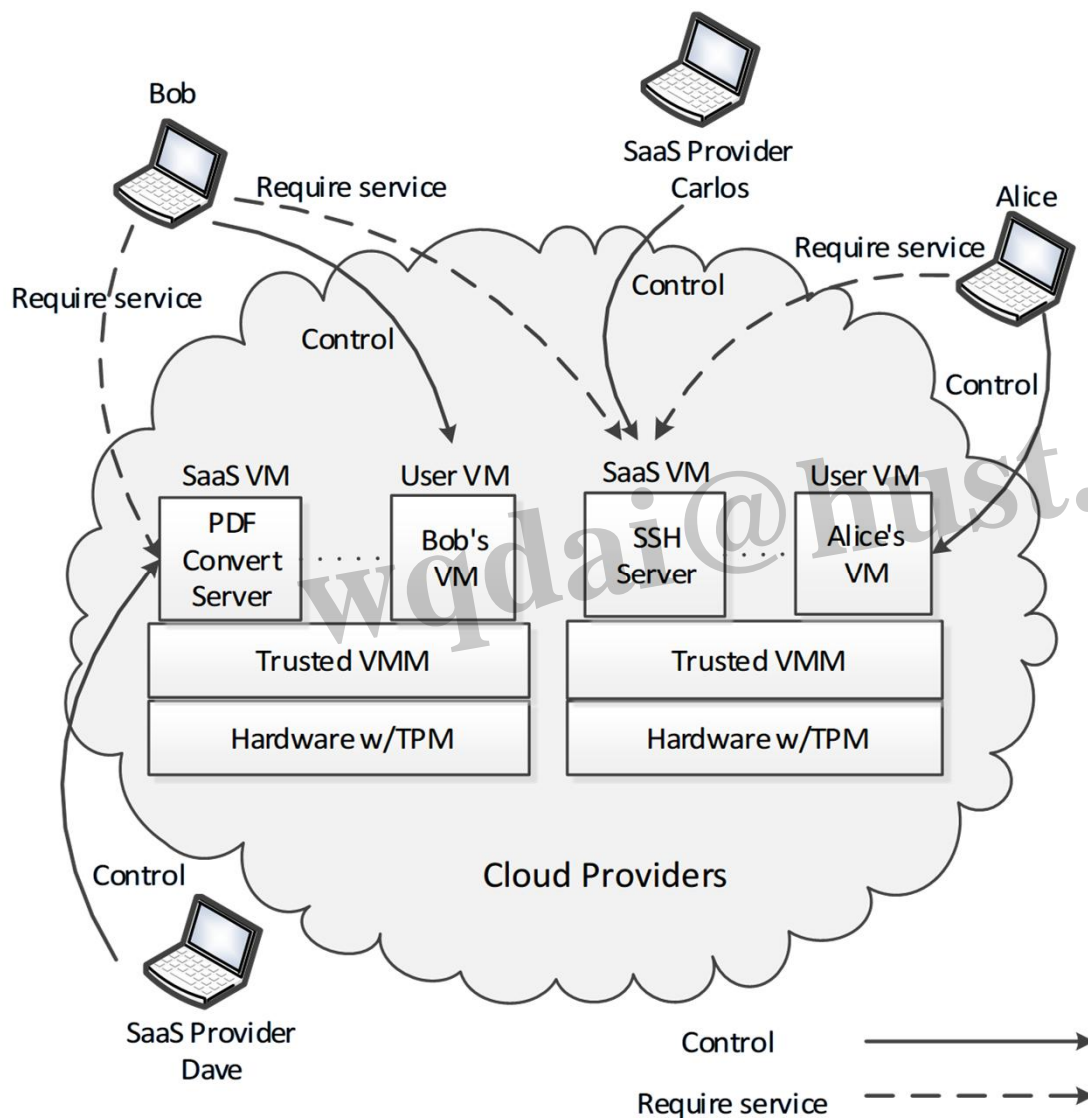
- 可信计算要求安全状态不可逆转，而云平台支持虚拟机快照与回滚的特点与可信计算相违背，比如一个虚拟机从安全状态回滚到不安全状态，从而导致安全状态不一致。因此如何消除虚拟机回滚造成的影响是一个重要的安全挑战。
- 现有的虚拟化可信平台芯片（vTPM）并不支持回滚，无法在虚拟机回滚的同时将自身状态回滚到快照时刻的状态。
- 简单的回滚vTPM数据，也会造成新产生数据的丢失，并且会带来重放攻击。

云环境场景分析-安全状态不一致



SaaS 服务商 Carlos 提供 SSH 服务给云用户 Alice 前，正常启动 SaaS VM（处于状态 S_{trusted} ），Alice 在远程验证后信任该服务，之后 Carlos 再回滚虚拟机到包含恶意代码的状态 ($S_{\text{untrusted}}$)，由于现有的 vTPM 不会回滚，因此无论 Alice 何时验证，VM 都将属于可信状态。

云环境场景分析

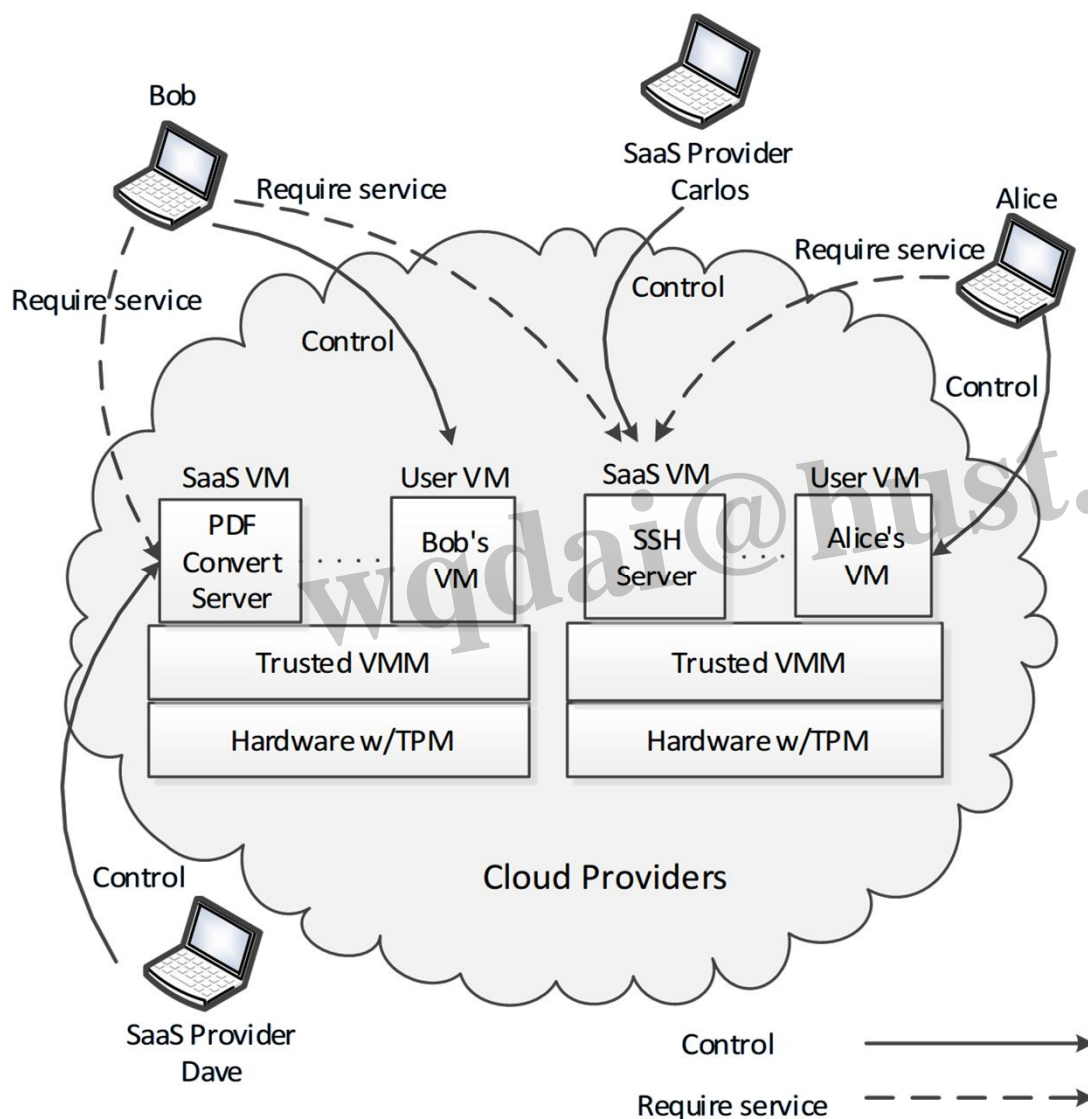


Seal Storage

VM回滚了，PCR没有回滚导致seal的密钥无法解封。

应用程序不支持回滚

云环境场景分析-重放攻击



1.另外简单的回滚vTPM同样会导致重放攻击：尽管vTPM在虚拟机回滚到不可信状态(Suntrusted)也同样回滚，但如果服务商在Alice每次做远程证明的时候都回滚虚拟机到可信状态，则可以利用重放攻击消除入侵痕迹。

2.此外像pdf转换服务提供商Dave使用能免费使用10次的pdf转换软件为Bob提供服务，即使是pdf软件借助vTPM记录实际使用次数，由于Dave在回滚虚拟机的时候vTPM的数据也发生回滚保持一致性，从而导致服务商可以无限次使用免费软件给其他云用户收费服务。

虚拟机安全回滚机制

- 解决思路:

- 首先为了保证虚拟机安全状态一致性, 需要实现在对虚拟机做快照时, 也对vTPM的状态做了快照, 保证可信执行环境的状态与vTPM的状态一一对应。
- 为了保证VM在虚拟机在做过一次快照后新增的密钥等安全数据不在回滚过程中发生丢失现象, 需要在vTPM回滚后恢复密钥数据等等。
- 为了解决回滚虚拟机带来的重放攻击问题, 在每次创建快照(回滚)时会把快照时间(回滚时间)、执行创建命令的用户信息(执行回滚命令的用户信息)、vTPM当前状态(vTPM回滚前的状态)以及虚拟机当前状态(虚拟机回滚前状态)等信息的度量值分别扩展到两组新增的PCRs中去, 从而保证了快照的唯一性, 用永远不会发生回滚操作的PCRs来真实地反映了虚拟机的回滚轨迹, 使得攻击者无法通过回滚虚拟机来消除攻击痕迹。

虚拟机安全回滚机制

- 解决思路:

$$PCR_{24}^n = SHA-1(PCR_{24}^{n-1} | SHA-1(snap_time))$$

$$PCR_{28}^n = SHA-1(PCR_{28}^{n-1} | SHA-1(roll_time))$$

$$PCR_{25}^n = SHA-1(PCR_{25}^{n-1} | SHA-1(user_UID))$$

$$PCR_{29}^n = SHA-1(PCR_{29}^{n-1} | SHA-1(user_UID))$$

$$PCR_{26}^n = SHA-1(PCR_{26}^{n-1} | SHA-1(vTPM_snap))$$

$$PCR_{30}^n = SHA-1(PCR_{30}^{n-1} | SHA-1(vTPM_snap))$$

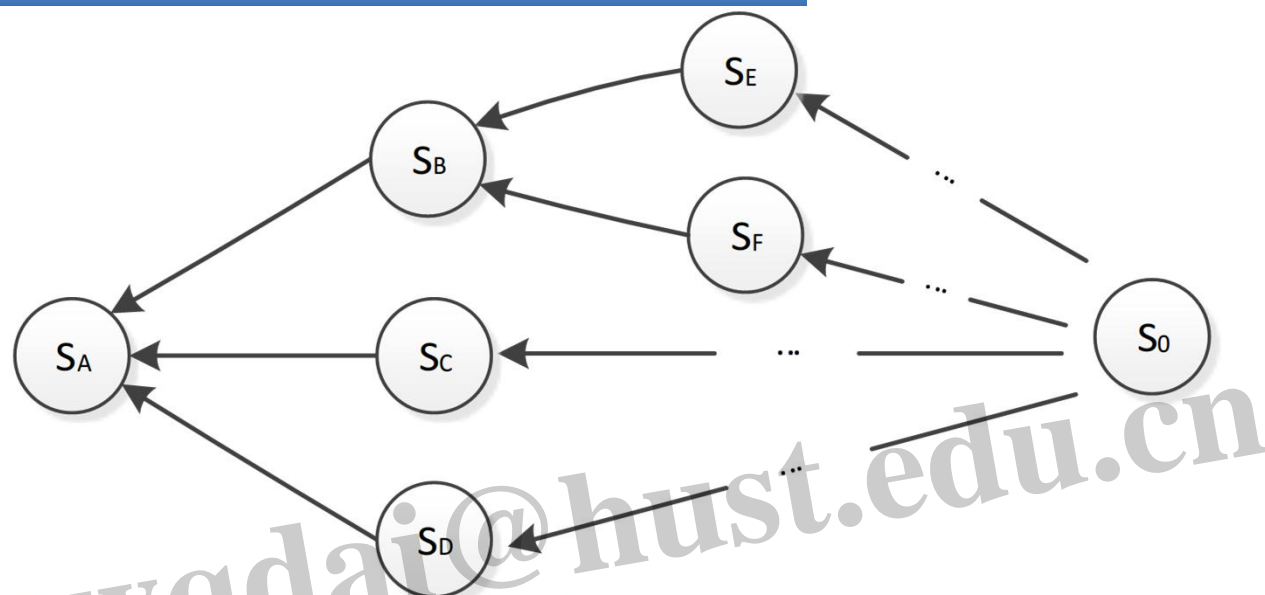
$$PCR_{27}^n = SHA-1(PCR_{27}^{n-1} | SHA-1(app_event))$$

$$PCR_{28 \sim 31}^n \neq 0$$

Table 2: New added PCRs

PCRs	Extend info	Can be reset or not	Can be extend or not
PCR_{24}	System time shows when take the snapshot	Can, by rvTPM in rvTPM	Can, by rvTPM in rvTPM
PCR_{25}	User UID shows who make the snapshot	Can, by rvTPM in rvTPM	Can, by rvTPM in rvTPM
PCR_{26}	Measurement of the vTPM snapshot	Can, by rvTPM in rvTPM	Can, by rvTPM in rvTPM
PCR_{27}	Optional, for applications use	Can, by rvTPM in rvTPM	Can be extend by applications
PCR_{28}	User UID shows who rollback the VM	Cannot	Can, by rvTPM in rvTPM
PCR_{29}	System time shows when VM rollback	Cannot	Can, by rvTPM in rvTPM
PCR_{30}	Measurement of the vTPM last snapshot	Cannot	Can, by rvTPM in rvTPM
PCR_{31}	Optional, for applications use	Cannot	Can be extend by applications

虚拟机生命周期



虚拟机生命周期的状态变迁记录:

由于记录回滚事件的PCR28~31是属于永远都不会被重置或者回滚的PCRs，因此同样是虚拟机处于 S_A 状态，无论是经过 S_E 到 S_B 再到 S_A 还是 S_C 到 S_A ，验证者都能通过这几个PCR值，发现虚拟机是否曾经被攻击过（在 S_E 状态时被窃取了隐私）或者被偷偷缩减计算资源（在 S_F 状态时内存被减少了一半）或者是软件试用到期（在 S_C 状态时软件已经达到试用次数）等等。

虚拟机生命周期

$$PCR_{24\sim31}^0 = 0$$

After the user take the snapshot S_1 (snap_time, user_UID, vTPM_snap), then the

$$\begin{aligned} PCR_{24\sim26}^1 &= SHA-1(PCR_{24\sim26}^0 | SHA-1(S_1)) \\ &= SHA-1(0 | SHA-1(S_1)) \end{aligned}$$

Then every time when the VM take the snapshot S_n (snap_time, user_UID, vTPM_snap, $n>0$), the rvTPM will first reset the $PCR_{24\sim27}$,

$$PCR_{24\sim27}^{n-1} = 0$$

then extend them with the information of the snapshot S_n :

$$\begin{aligned} PCR_{24\sim26}^n &= SHA-1(PCR_{24\sim26}^{n-1} | SHA-1(S_n)) \\ &= SHA-1(0 | SHA-1(S_n)) \end{aligned}$$

When the first time the VM rollback from the state S_B to snapshot S_A : Our rvTPM will restore the vTPM including the newly added $PCR_{24\sim27}$, then the remote can get the snapshot information from these PCRs. So the

$$PCR_{24\sim26}^A = SHA-1(0 | SHA-1(S_A))$$

The second group of PCRs need to record the rollback event, it will be extended with the rollback information, so the

$$\begin{aligned} PCR_{28\sim30}^1 &= SHA-1(PCR_{28\sim30}^0 | SHA-1(S_B)) \\ &= SHA-1(0 | SHA-1(S_B)) \end{aligned}$$

$PCR_{28\sim30}$ stand for the birth state to the last state before the rollback event and the $PCR_{24\sim26}$ stand for the state revert to. So here the remote verifier can find the VM is from new one to state S_B then rollback to state S_A .

After then, when the VM running to another state S_C and then roll back to snapshot S_A : Our rvTPM will restore the vTPM including the newly added $PCR_{24\sim26}$.

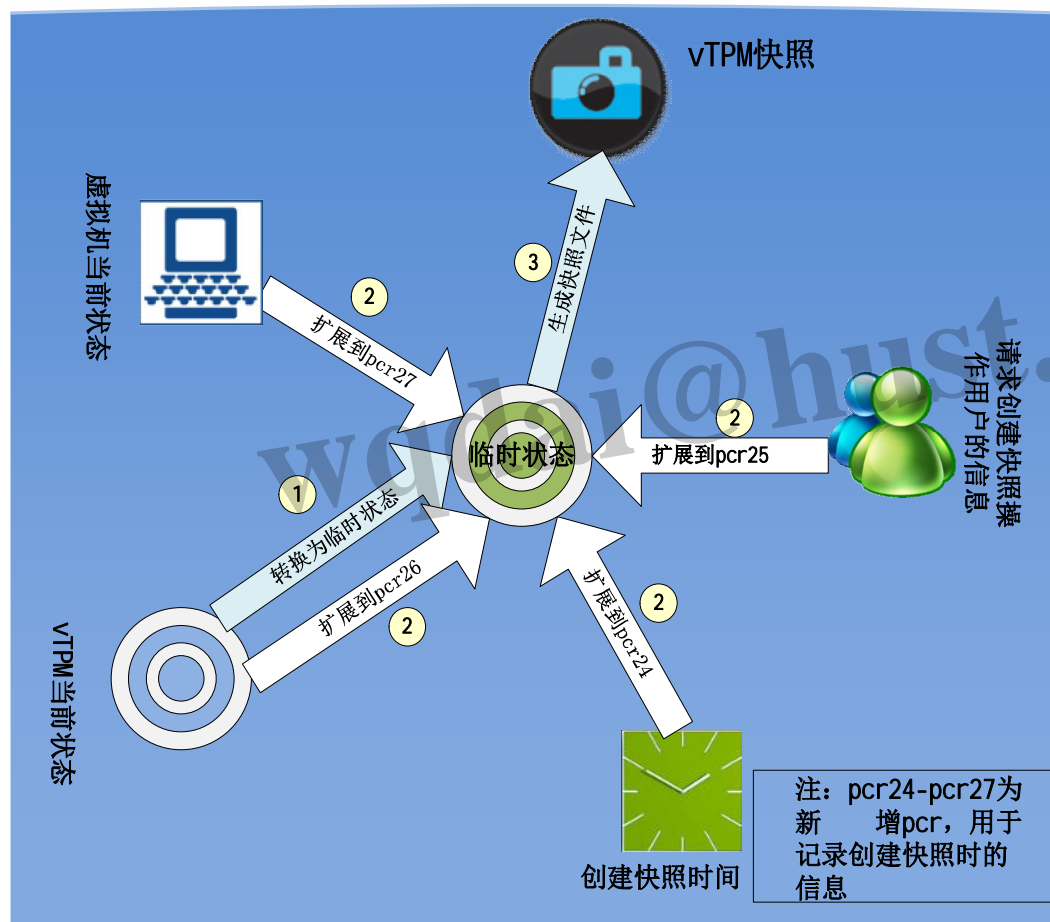
$$PCR_{24\sim26}^A = SHA-1(0 | SHA-1(S_A))$$

And then extend the $PCR_{28\sim30}$

$$\begin{aligned} PCR_{28\sim30}^2 &= SHA-1(PCR_{28\sim30}^1 | SHA-1(S_C)) \\ &= SHA-1(SHA-1(0 | SHA-1(S_B)) | SHA-1(S_C)) \end{aligned}$$

虚拟机创建快照

创建vTPM快照时

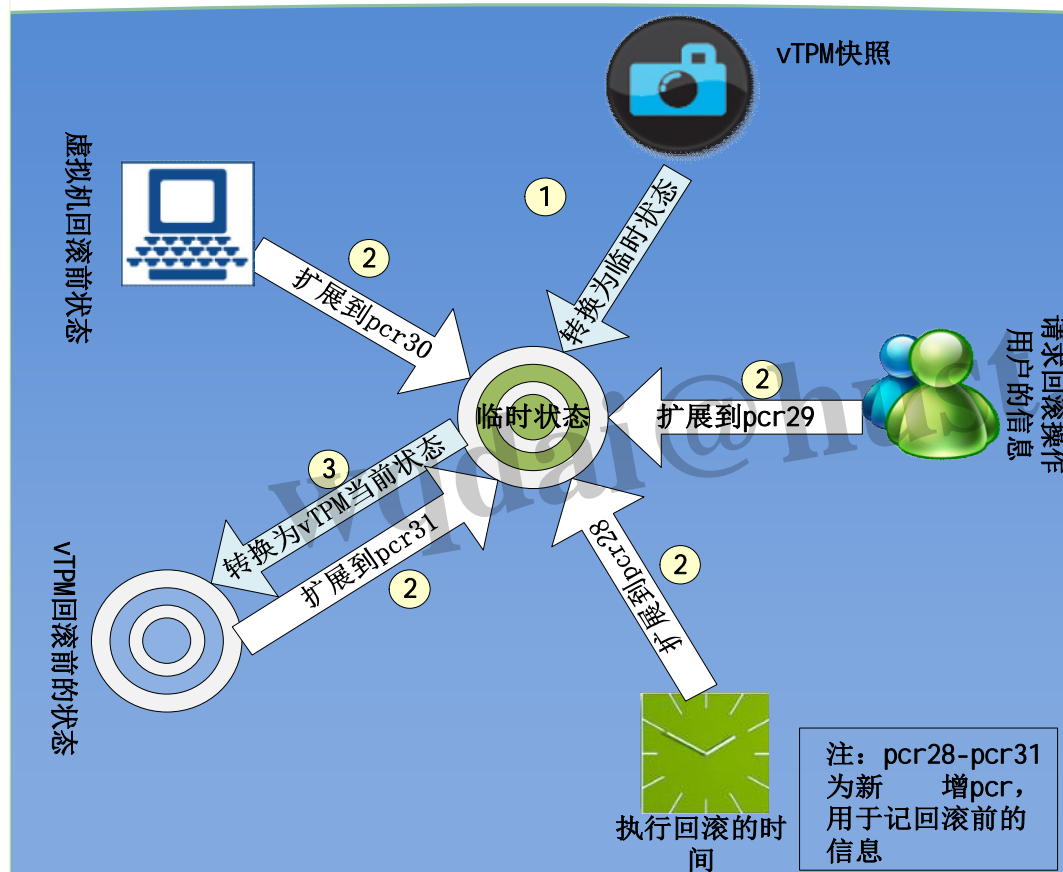


创建vTPM快照:

1. 将vTPM的当前状态转换为临时状态
2. 把生成快照的时间、请求创建快照操作用户信息、vTPM当前状态以及虚拟机的当前状态扩展到 pcr24-pcr27中
3. 最后，把临时状态转换为vTPM快照文件。

虚拟机安全回滚

vTPM快照回滚



vTPM快照回滚：

1. 将vTPM的当前状态转换为临时状态
2. 把回滚时间、执行回滚命令的用户信息、虚拟机回滚前的状态以及vTPM回滚前的状态扩展到 pcr28-pcr31 中
3. 最后，把临时状态转换为vTPM快照文件

解决方案（虚拟机安全回滚机制续）

