身份认证协议描述

初始化阶段

在开始使用之前，分别在功能平台，认证平台以及与认证平台直接交互的机密存储区进行函数写入和其他初始化工作，具体初始化内容如下：

系统管理员为各个功能平台分配唯一身份标识 ID_j，在统一身份认证内部生成其唯一的 RSA 公钥体制的公钥 k_p 和私钥 k_v，在机密计算内部的安全区使用基于硬件芯片熵池的 RDRAND 指令生成真随机数作为唯一的安全区机密存储的主密钥 K_{UCP}，并对统一认证平台分配唯一的 ID_{UCP}。

通信建立阶段

该阶段在认证平台端和用户端之间建立可靠通信连接，基于公钥密码体制完成对称密钥 SK 的分配过程。本文考虑到实际应用场景的便捷性，不引入智能卡进行本地存储，无法进行本地身份校验，也无法隐藏对称密钥。因此当用户执行注册，登陆与密钥协商阶段前，需要在用户端和认证平台端之间建立可靠通信连接。具体流程如下图所示：

1） 用户端生成随机数 r_1，获取当前时间戳 TS_1。

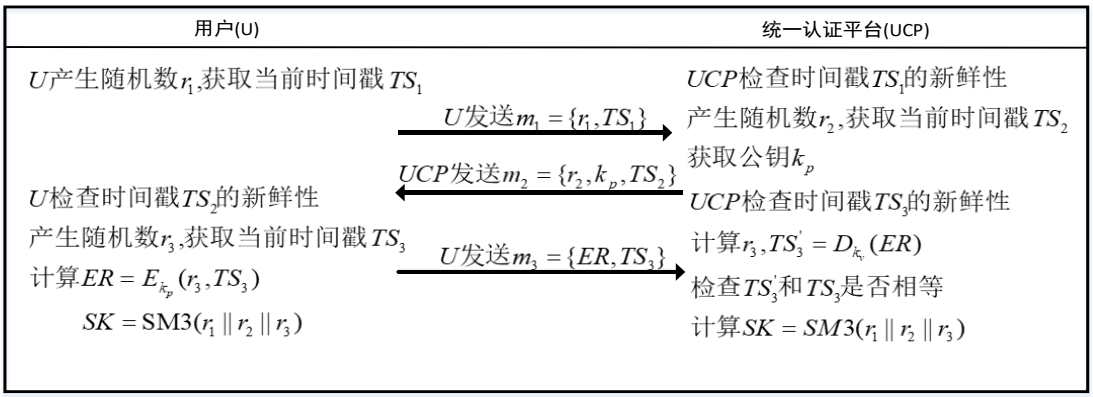
2） 将 $m_1=\{r_1,TS_1\}$ 以 POST 请求方式发送给认证平台端。

3） 认证平台端接收到 m_1 后，验证时间戳 TS_1 是否有效（有效的标志为当前时间戳与待验证的时间戳之差小于通信传播的最大时间传输延迟，下同），若无效则拒绝建立通信，若有效则生成随机数 r_2 并获取当前时间戳 TS_2，获取认证平台的 RSA 公钥 k_p。

4） 将 $m_2=\{r_2,k_p,TS_2\}$ 以 POST 请求方式返回给用户端。

5） 用户端接收到 m_2 后，验证时间戳 TS_2 是否有效，若有效则生成随机数 r_3 并获取当前时间戳 TS_3，使用获取到的 m_2 中的公钥 k_p 将 r_3, TS_3 加密成 ER，将 $m_3=\{ER,TS_3\}$ 以 POST 请求方式发送给认证平台端。之后在用户端进行 $r_1||r_2||r_3$ 操作，并将操作结果进行单向哈希，将哈希结果作为对称密钥 SK 保存在用户端。

6） 认证平台端接收到 m_3 后，验证时间戳 TS_3 是否有效，若无效则拒绝建立通信，若有效则使用 RSA 私钥 k_v 对 ER 进行解密，解密结果记作 r_3, TS_3'。检验 TS_3' 是否等于 TS_3。若相等，则执行 $r_1 \| r_2 \| r_3$ 操作，并将操作结果进行单向哈希，将哈希输出结果作为对称密钥 SK 保存在认证平台端。



用户注册阶段

在该阶段，用户通过前一阶段建立的可靠通信连接向统一认证平台发起注册。用户在注册时输入登录 ID(ID_i)以及登录密码(PW_i)，之后发送给统一认证平台端，由统一认证平台认证后计算出用户假名 PID_i 以及验证信息 V 并进行保存。具体流程如下图所示：

1） 用户输入 ID_i 及 PW_i ，并由前端确定输入符合规范后进行 $ID_i \| PW_i$ 操作，将操作后得到的结果进行哈希并存储到 UID_i 中。之后获取最新的时间戳 TS_1 ，根据通信建立阶段计算出的对称密钥 SK 对 UID_i 和 TS_1 进行加密并保存到 EID_i 中。

2） 将 $m_1 = \{EID_i, TS_1\}$ 以 POST 请求方式发送给认证平台端。

3） 认证平台在接收到用户端发来的 m_1 后验证时间戳 TS_1 是否有效。若无效则拒绝用户注册请求。若有效则根据对称密钥 SK 将接收到的 m_1 进行解密，解密后的 EID_i 和 TS_1 对应保存到 UID_i 和 $TS_1^{'}$ 中。之后验证 $TS_1^{'}$ 和 TS_1 是否相等，若不相等则拒绝用户注册请求，相等则将 UID_i 通过机密计算环境的非信任区对外接口发送到非信任区中。之后调用安全区 ecall 函数在安全区内部将 UID_i 和 K_{UCP} 进行拼接并进行单向哈希，哈希的结果保存在 PID_i 中，并通过对外非信任接口返回。之后验证返回的 PID_i 是否是 $PIDlist$ 中的元素。若属于则拒绝用户再次注册请求，若不属于则将 PID_i 添加到 $PIDlist$ 中。获取统一认证平台的 ID(ID_{UCP})，并产生最新的时间戳 TS_2 ，之后使用对称密钥 SK 对 TS_2 进行加密保存到应答序列号 ACK_{UCP} 中，通过可靠通信信道将 $m_2 = \{ACK_{UCP}, TS_2\}$ 发送给用户端。之后进行 $ID_{UCP} \| PID_i$ 操作，将操作得到的结果进行哈希后保存到验证信息 V 中，将键值对 $\{PID_i - V\}$ 进行保存。

4） 用户端在接收到 m_2 后检查 TS_2 是否有效，若有效则使用 SK 对接收到的应答序列号 ACK_{UCP} 进行解密，将解密结果保存到 $TS_2^{'}$ 中，之后验证 $TS_2^{'}$ 是否等于 TS_2 ，若不相等则返回第一步再次发送注册请求，若相等则用户端确定注册阶段完成，注册成功。

用户(U)		统一认证平台(UCP)
U输入 ID_i, PW_i		UCP检查时间戳TS_1的新鲜性
计算$UID_i = SM3(ID_i \| PW_i)$		计算$UID_i, TS_1^{'} = D_{SK}(EID_i)$
获取当前时间戳TS_1		检查$TS_1^{'}$和TS_1是否相等
计算$EID_i = E_{SK}(UID_i, TS_1)$		将UID_i传送到SGX的安全区
	U发送$m_1 = \{EID_i, TS_1\}$ →	在安全区内计算$PID_i = SM3(UID_i \| K_{UCP})$
		从安全区中返回PID_i
		检查$PIDlist$是否包含PID_i
		获取ID_{UCP}
U检查时间戳TS_2的新鲜性		获取当前时间戳TS_2
计算$TS_2^{'} = D_{SK}(ACK_{UCP})$	← UCP发送$m_2 = \{ACK_{UCP}, TS_2\}$	计算$V = SM3(ID_{UCP} \| PID_i)$
检查$TS_2^{'}$和TS_2是否相等		$ACK_{UCP} = E_{SK}(TS_2)$
		存储键值对$\{PID_i - V\}$

功能平台注册阶段

该阶段默认在可信环境中运行，功能平台和认证平台互相预埋信息，在功能平台端存储

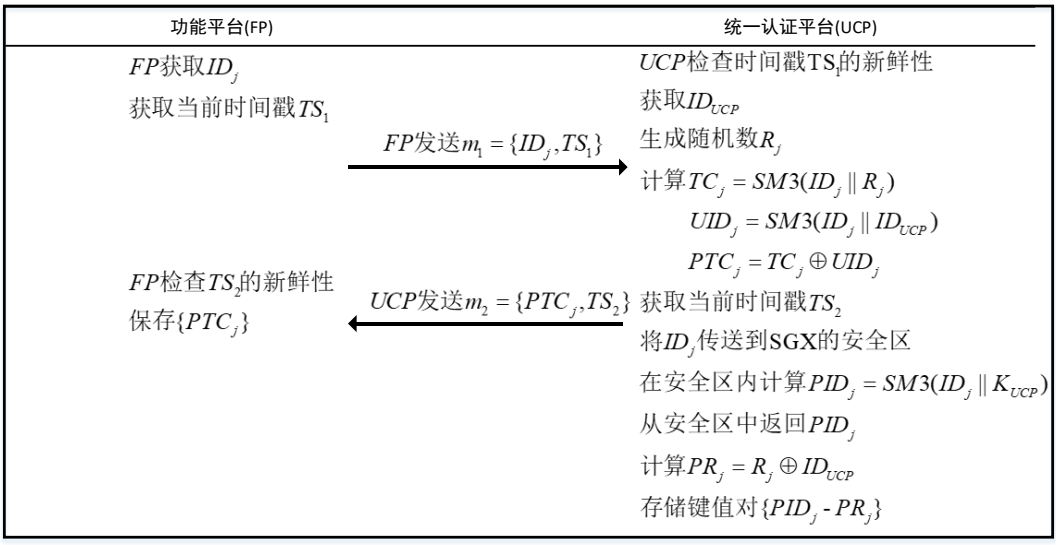
由功能平台 ID(ID_j)认证平台 ID(ID_{UCP})，以及在认证平台端生成的随机数通过一系列运算得到的PTC_j，而认证平台端存储功能平台的假名PID_j以及用于加密的随机数PR_j。具体步骤如下图所示：

1） 功能平台端获取其唯一身份标识 ID(ID_j)，并获取最新时间戳TS_1。

2） 功能平台在可靠环境内将$m_1=\{ID_j,TS_1\}$发送给统一认证平台。

3） 统一认证平台在接收到m_1后，首先检查时间戳是否有效，若无效则拒绝功能平台注册请求，若有效则获取统一认证平台唯一身份标识 ID(ID_{UCP})，之后产生随机数R_j。接下来，执行$ID_j \| R_j$操作，并将结果进行哈希后存储到TC_j中；进行$ID_j \| ID_{UCP}$操作，并将操作结果进行哈希后存储到UID_j中；之后将TC_j 和UID_j进行异或，结果保存到PTC_j中。完成上述操作后，统一认证平台端获取最新时间戳TS_2。之后，统一认证平台在可靠环境内将$m_2=\{PTC_j,TS_2\}$发送给发出注册请求的功能平台。之后通过机密计算环境的非信任区对外接口将ID_j发送到非信任区中，调用 ecall 函数在信任区内执行$ID_j \| K_{UCP}$操作，之后将连接结果进行哈希保存到PID_j中。下一步将PID_j从信任区中返回出来，并将之前生成的随机数R_j和ID_{UCP}做异或，结果保存到PR_j 中。最后将键值对$\{PID_j - PR_j\}$进行保存。

4） 功能平台端在接收到m_2后检查TS_2是否有效，若无效，则返回该阶段第一步；若有效，则将PTC_j进行保存，功能平台注册成功。



登录和密钥协商阶段

该阶段在可靠环境中运行，在上述用户端和功能平台端分别向认证平台端进行注册后，使用户端和功能平台端共享随机数N_i，N_j并以此计算出仅由用户端和功能平台端持有的会话密钥KEY_{ij}。其中 Hmac 和 Ver 是一对哈希消息码加密和验证的函数，具体流程如下图所示：

1） 用户端输入身份标识ID_i,口令PW_i以及访问目标功能平台身份表示ID_j。生成随机数N_i,获取当前时间戳TS_1，执行$ID_i \| PW_i$操作，将操作结果哈希后保存在变量UID中。

用户端使用在通信建立阶段协商的对称密钥SK，分别对UID_i，ID_j，N_i,TS_1加密并将加密结果保存为EID_i。

2） 用户端将$m_1=\{EID_i,TS_1\}$以 POST 请求方式发送给认证平台端。

3） 认证平台端接收到m_1后，验证TS_1的有效性，若无效则拒绝登录请求，若有效则使用认证平台端在通信建立阶段协商的对称密钥SK对EID_i解密以获取UID_i,ID_j,N_i ,$TS_1^{'}$。检验$TS_1^{'}$,TS_1是否相等，若不相等则拒绝登录请求，若相等则将UID_i通过机密计算环境非信任区对外接口传送到非信任区中。在非信任区通过调用 ecall 在安全区内执行$UID_i \| K_{UCP}$操作并对其结果进行单向哈希，结果保存在PID_i并通过非信任区对外接口传送给认证平台端。认证平台端通过验证返回的PID_i是否是$PIDlist$的元素，确定用户是否已注册。若用户未注册，则拒绝登录请求；若用户已注册，则凭借PID_i获取认证平台端标识ID_{UCP}和验证信息V。执行$ID_{UCP} \| PID_i$操作并对其结果哈希，将哈希结果$V^{'}$和V比对，若不相等则表明用户口令更改，若相等则将ID_j 通过机密计算环境非信任区对外接口传送到非信任区中，执行$ID_j \| K_{UCP}$操作并对其结果哈希，将哈希结果作为PID_j 并通过非信任区对外接口传送给认证平台端。认证平台端凭借PID_j查找PR_j ，将PR_j和ID_{UCP}做异或，结果保存在R_j中。执行$ID_j \| R_j$操作，并对其进行哈希，结果保存在TC_j 中。同样执行$ID_j \| ID_{UCP}$操作，将对其哈希的结果保存在UID_j中。获取当前时间戳TS_2，执行$TC_j \| ID_j \| TS_2$操作，并对操作结果进行哈希，哈希结果和N_i异或后存储为TRN_1。之后执行$TRN_1 \| TC_j \| UID_j \| TS_2$操作，使用$N_i$作为密钥生成上述操作结果的 Hmac（消息认证码）q_1。

4） 认证平台端将$m_2=\{UID_j,TRN_1,q_1,TS_2\}$以 POST 请求方式发送给功能平台端。

5） 功能平台端收到m_2后，验证时间戳TS_2的有效性。若无效，则拒绝认证平台端的访问，若有效，则功能平台获取自身的ID_j以及对应的PTC_j。之后将PTC_j与UID_j进行异或，异或的结果保存到TC_j中；并进行$TC_j \| ID_j \| TS_2$操作，将操作得到的结果哈希后与进行异或并将结果保存到N_i中。运算完成后，根据验证函数验证q_1是否等于$TRN_1 \| TC_j \| UID_j \| TS_2$，若不相等则拒绝认证平台端的访问。若相等则生成随机数N_j并获取最新时间戳TS_3。之后进行$N_i \| TC_j \| ID_j \| TS_3$操作，将操作结果进行哈希并和$N_j$进行异或，将结果保存到$TRN_2$中。完成计算后进行$TRN_2 \| N_i \| TC_j \| UID_j \| TS_3$，并使用$N_j$作为密钥生成上述操作结果的 Hmac（消息认证码）q_2。

6） 功能平台端通过可靠通信信道将$m_3=\{TRN_2,q_2,TS_3\}$发送给统一认证平台端。发送后进行$N_i \| N_j$操作，并将结果进行哈希作为对称密钥KEY_{ij}用于后续加密用户请求。

7） 统一认证平台端在接收到m_3后验证TS_3是否有效，若无效则再次发送m_2，有效则进行$N_i \| TC_j \| ID_j \| TS_3$并将结果进行哈希后与$TRN_2$进行异或并保存到$N_j$中。之后验证通过验证函数验证$TRN_2 \| N_i \| TC_j \| UID_j \| TS_3$是否等于$q_2$，若相等则获取最新时间戳$TS_4$，并通过对称密钥$SK$对$N_j$和$TS_4$进行加密，加密后的结果保存到$EN_j$中。

8） 统一认证平台端将$m_4=\{EN_j,TS_4\}$发送给用户端。

9） 用户端在接收到后首先验证TS_4是否有效，若有效则根据SK对m_4进行解密，得到解密后的时间戳$TS_4^{'}$以及认证平台端生成的随机数N_j。之后验证$TS_4^{'}$是否等于TS_4，若相等

则进行 $N_i \| N_j$ 操作，并将得到的结果进行哈希得到对称密钥 KEY_{ij} 。