ICS 35.040
CCS L 80

GM

中华人民共和国密码行业标准

GM/T 0079—2020

可信计算平台直接匿名证明规范

Direct anonymous attestation specification for trusted computing platform

2020-12-28 发布　　2021-07-01 实施

国家密码管理局　　发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由密码行业标准化技术委员会提出并归口。

本文件起草单位：中国科学院软件研究所、国民技术股份有限公司、清华同方股份有限公司、北京华电卓识信息安全测评技术中心有限公司、兴唐通信科技有限公司。

本文件主要起草人：冯登国、秦宇、初晓博、张振峰、冯伟、赵世军、陈小峰、奚璨、杨糠、汪丹、刘鑫、郑必可、刘峰、张倩颖、常德显、刘韧、吴秋新、邵健雄、王微谨、杨波、张英骏。

可信计算平台直接匿名证明规范

1 范围

本文件规定了可信计算平台的直接匿名证明协议的功能、接口和数据结构。

本文件适用于可信计算平台直接匿名证明协议应用、匿名证明服务和匿名证明系统研发。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 32918—2016(所有部分) 信息安全技术 SM2 椭圆曲线公钥密码算法

GM/T 0012 可信密码模块接口规范

GM/Z 4001 密码术语

3 术语和定义

GM/Z 4001 界定的以及下列术语适用于本文件。

3.1

可信密码模块 trusted cryptography module；TCM

构建可信计算平台的基础硬件模块，为可信计算平台提供密码运算功能，具有受保护的存储空间。

3.2

签署密钥 endorsement key；EK

可信密码模块内部用于标识自身身份的密钥对，其用途只能用于加解密。根据上下文的不同情境，这个术语可能代表一个密钥对、密钥对中的公钥或者代表密钥对中的私钥。

3.3

TCM 服务模块 TCM service module

可信密码模块向应用程序提供服务的软件中间件。

3.4

直接匿名证明 direct anonymous attestation；DAA

可信计算平台所使用的匿名身份鉴别方案。

3.5

基于椭圆曲线的直接匿名证明 elliptic curve-based direct anonymous attestation

基于椭圆曲线密码学方案的直接匿名证明方案。

3.6

证明方 prover

直接匿名证明中的证明方，一般包含主机和可信密码模块两个部分。

3.7

主机 host

直接匿名证明中证明方拥有的内嵌可信密码模块的安全主机。

3.8

凭证颁发方　issuer

直接匿名证明中，为可信密码模块颁发凭证的参与方。

3.9

验证方　verifier

直接匿名证明中，验证远程可信密码模块身份的参与方。

4　符号与缩略语

4.1　符号

GB/T 32918—2016(所有部分)中界定的以及下列密码学符号适用于本文件。

0：整数 0、比特 0 或者有限域加法单位元。

1：整数 1、比特 1 或者有限域乘法单位元。

a，b：F_q 中的元素，它们定义 F_q 上的椭圆曲线 E。

e：$G_1 \times G_2 \rightarrow G_T$：双线性映射，将($G_1$，$G_2$)中元素映射至 G_T 中元素。

exp(l，m)：有限域元素 l 的 m 次幂，也记作 l^m。

E：有限域上由 a，b 定义的一条椭圆曲线。

$E(F_q)$：E 中所有坐标属于 F_q 的点(包括无穷远点 O)所构成的集合。

F_q：q 阶有限素域。

F_{q^k}：q^k 阶有限域，q 阶有限域的扩域。

G_n：椭圆曲线的一个基点，其阶为素数，下标 n 是整数，用为于区分不同基点。

G_T：有限域的一个基点，其阶为素数。

$l+m$：有限域元素 l 与 m 的域加法运算结果。

$l \times m$：有限域元素 l 与 m 的域乘法运算结果，在不引起歧义的情况下，也记作 lm。

P：$P=(x_p, y_p)$是椭圆曲线上除零点 O 外的一个点，其坐标满足椭圆曲线方程。

P_1+P_2：椭圆曲线 E 上两个点 P_1 与 P_2 加法运算，也可记作 $P_1 g P_2$。

O：椭圆曲线上一个特殊点，称为无穷远点或零点，是椭圆曲线加法群的单位元。

x_p：点 P 的 x 坐标。

y_p：点 P 的 y 坐标。

Z_n：整数模素数 n 的剩余类。

$\langle g \rangle$：g 生成的循环群。

$\langle g_T \rangle$：g_T 生成的循环群。

$[k]P$：椭圆曲线上点 P 的 k 倍点，不引起歧义的情况下，也记作 P^k。

$\sharp E(F_q)$：$E(F_q)$中点的数目，又称为 $E(F_q)$的阶。

$g \in_R G$：从集合 G 中随机选择元素 g。

$HASH$：标准密码杂凑函数。

4.2　缩略语

下列缩略语适用于本文件。

TCM：可信密码模块(Trusted Cryptography Module)

EK：签署密钥(Endorsement Key)

TSM：TCM 服务模块(TCM Service Module)

DAA：直接匿名证明(Direct Anonymous Attestation)

ECDAA：基于椭圆曲线的直接匿名证明(Elliptic Curve-based DAA)

5 密码算法

本文件采用国家密码管理主管部门认可的密码算法。

6 直接匿名证明功能

6.1 概述

直接匿名证明用于 TCM 安全芯片的匿名身份证明。功能模型说明了系统的参与方的构成、作用与目标，以及直接匿名证明的整体流程。功能算法说明了实现模型规定各方所必须执行的原子算法以及有关的参数选择等。

6.2 模型

6.2.1 系统组成

ECDAA 系统主要由凭证颁发方(Issuer)、证明方(Prover)和验证方(Verifier)三方面的参与者构成见图 1，其中证明方根据 ECDAA 计算位置不同而分为主机和 TCM 安全芯片，两者协同计算共同完成了匿名凭证申请和匿名证明过程。

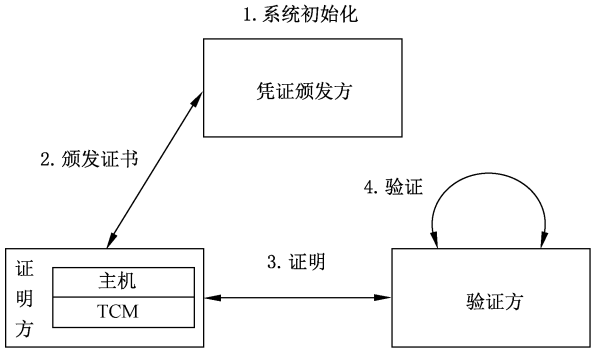

图 1 系统组成和基本流程

在 ECDAA 系统中，凭证颁发方负责初始化 ECDAA 系统参数，为 TCM 安全芯片颁发 ECDAA 凭证，验证 TCM 安全芯片的身份是否已经被撤销。一般而言，TCM 安全芯片的生产厂商可以作为凭证颁发方，对于不同厂商的 TCM 芯片可以选择不同的 ECDAA 系统参数。也可以采用一个独立的权威机构充当凭证颁发方，集中式管理 TCM 匿名凭证。

证明方平台是硬件上嵌入 TCM 安全芯片，支持 ECDAA 规范的安全 PC、笔记本等可信计算平台。证明方平台主要功能是 TCM 匿名凭证的申请、TCM 匿名身份的证明，证明方平台驱动 TCM 通过执行 TCM_ECDAA_Join 命令和相关主机计算向凭证颁发方请求匿名身份凭证；证明方平台执行 TCM_ECDAA_Sign 命令和相关主机计算向验证方平台匿名证明 TCM 数字身份。

验证方平台主要是通过验证证明方平台提供的证明数据，认证证明方平台 TCM 身份，保证证明方平台的确是采用安全芯片 TCM 作为平台的身份。在验证 TCM 匿名身份的同时，需要向凭证颁发方请求验证 TCM 数字身份是否已经被撤销。

在 ECDAA 系统中，TCM 安全芯片的匿名身份私钥 f 只允许保存在 TCM 芯片内部，不允许被导出。TCM 的匿名身份私钥和匿名证明凭证可以存在多个，但是推荐只使用一个匿名身份私钥和凭证。TCM 匿名证明过程(包括证明和验证)只有 TCM 所有者才能执行，并且只有 TCM 所有者才能够清除不安全的匿名私钥。TCM 匿名身份凭证可以保存在芯片外部的主机平台，或者其他存储设备中。

证明方平台的核心计算功能由 TCM 的 TCM_ECDAA_Join 和 TCM_ECDAA_Sign 命令来完成，应当只有较高的权限才能执行这些 ECDAA 命令。ECDAA 命令是非常耗费 TCM 和主机计算资源的命令，它需要大量的 TCM 芯片内部资源来完成一系列计算操作。在 TCM 安全芯片执行 ECDAA 命令过程中，需要禁止其他 TCM 命令操作执行。

6.2.2 基本流程

ECDAA 系统各个参与者之间主要通信流程包含如下步骤：

a) 系统初始化：设置 ECDAA 系统的公共参数，生成凭证颁发方用于颁发匿名凭证的公私钥对。
b) 凭证颁发：证明方向凭证颁发方申请和获得匿名凭证。
c) 证明：证明方利用匿名凭证以及对应私钥创建匿名证明数据。
d) 验证：验证方验证特定消息是由合法 TCM 正确签名的。

6.2.3 安全目标

ECDAA 系统主要解决的问题是可信计算平台/TCM 芯片用户如何向远程验证方证明自身平台的确使用可信密码模块 TCM，也即是 TCM 安全芯片如何认证自身的问题。在认证 TCM 身份的同时，还需要保护平台身份隐私，要求远程验证方无法知道 TCM 安全芯片确切的身份，无法链接多次 TCM 会话。为了满足上述安全需求，ECDAA 系统需要实现如下安全目标：

a) 不可伪造性：只有配备 TCM 芯片并依赖芯片申请了匿名身份凭证，证明方才能进行 ECDAA 匿名证明，其他任何攻击者都无法伪造 ECDAA 证明数据。
b) 匿名性：在 TCM 未被攻破的情况下，任意用户通过协议数据无法获得当前 TCM 的唯一性标识。
c) 不可关联性：验证方平台无法建立两个 ECDAA 证明会话间的关联性，也即是对于两个不同的证明会话，验证方平台无法判定是否来自同一个 TCM。
d) 恶意 TCM 检测：当 TCM 拥有的匿名凭证对应的私钥被泄露后，验证方及凭证颁发方在协议运行过程中可及时发现该类泄露。

6.3 算法

6.3.1 系统初始化 1

该算法用于凭证颁发方生成椭圆曲线系统公开参数和自身持有的秘密信息，其输入、输出和算法流程如下：

a) 输入：安全参数 1^l、凭证颁发方证书链根密钥公钥 k_0 和凭证颁发方公共参数签名密钥私钥 k_n^{-1}。
b) 输出：公共参数 $gpk=(q,a,b,p,g_1,g_2,e,h_1,h_2,w,H_1,H_2,H_3,H_4,T_1,T_2,T_3,T_w)$，凭证颁发方对公共参数的签名 $cre=sign_{k_n^{-1}}(issuerSettings)$ 和凭证颁发方秘密信息 $isk=r$。
c) 算法流程：
   1) 按照直接匿名证明系统参数 (q,a,b,g_1,g_2,p)。其中，a，b 和 F_q 共同定义了椭圆曲线 $E(F_q)$，g_1，g_2 分别是 $E(F_q)$ 的基点，其阶均为素数 p。
   2) 选择双线性映射运算 $e:G_1\times G_2\rightarrow G_T$。其中 G_1，G_2 分别是以 g_1，g_2 为生成元的循环群，其阶均为素数 p，G_T 是扩域 F_{q^k} 上以 g_T 为生成元的 p 阶循环群，k 为椭圆曲线的嵌入度。运算 e 应满足如下性质：
      ——对于所有的 $P\in G_1$，$Q\in G_2$，所有的 l，$m\in Z_n$，满足：$e(lP,mQ)=e(P,Q)^{lm}$；
      ——存在 $P\in G_1$，$Q\in G_2$，使得 $e(lP,mQ)\neq 1_{G_T}$；
      ——存在一种有效的算法计算 $e(P,Q)$。
   3) 选择 $r\in_R Z_p^*$ 以及 $h_1\in_R G_1$，$h_2\in_R G_1$，计算 $w=g_2^r$。

4) 选择杂凑函数 $H_1:\{0,1\}^*\rightarrow\{0,1\}^{2l}$、$H_2:\{0,1\}^{6\lambda}\rightarrow Z_p$、$H_3:\{0,1\}^*\rightarrow G_2$、$H_4:\{0,1\}^*\rightarrow Z_p$。

5) 计算双线性映射 $T_1=e(g_1,g_2)$、$T_2=e(h_1,g_2)$、$T_3=e(h_2,g_2)$和 $T_w=e(h_2,w)$。

6) 计算密码杂凑值 $H_p=HASH(p)$、$H_{h1}=HASH(h_1)$和 $H_{k0}=HASH(k_0)$，根据 H_p、H_{h1}和 H_{k0}生成 TCM_ECDAA_ISSUER 数据结构(按附录 A 定义)的 $issuerSettings=(H_p,H_{h1},H_{k0})$，并用 k_n^{-1} 对其签名生成 $cre=sign_{k_n^{-1}}(issuerSettings)$。

7) 输出系统公共参数 $gpk=(q,a,b,p,g_1,g_2,e,h_1,h_2,w,H_1,H_2,H_3,H_4,T_1,T_2,T_3,T_w)$，对公共参数的签名 $cre=sign_{k_n^{-1}}(issuerSettings)$和秘密信息 $isk=r$。

6.3.2 系统初始化 2

该算法用于证明方主机和 TCM 设置凭证颁发方生成的椭圆曲线系统公开参数(详见附录 B)，其输入、输出和算法流程如下：

a) 输入：凭证颁发方公钥证书链，凭证颁发方验证的公共参数 $issuerSettings=(HASH(p),HASH(h_1),HASH(k_0))$，对公共参数的签名 $cre=sign_{k_n^{-1}}(issuerSettings)$。

b) 输出：若设置成功，则输出“有效”；否则输出“无效”。

c) 算法流程：

1) 证明方调用 TCM 的直接匿名证明功能接口 TCM_ECDAA_Setup(详见第 8 章)。具体地，TCM 验证凭证颁发方公钥证书链和凭证颁发方对公共参数的签名 cre，然后根据 issuerSettings设置 TCM 内部凭证颁发方信息 ECDAA_TCMSpecific ->ECDAA_digestIssuer(按附录 A 定义)。

2) 证明方主机将 gpk 可信存储。

6.3.3 凭证颁发算法 1

该算法用于证明方生成申请匿名凭证所需的信息，其输入、输出和算法流程如下：

a) 输入：凭证颁发方生成的挑战随机数 $n_I\in\{0,1\}^{2\lambda}$以及公共参数 gpk。

b) 输出：凭证申请信息 $comm$ 和证明所需的证明信息 aux。

c) 算法流程：

1) TCM 首先随机选择 $f\in Z_p{}^*$ 和 $r_f\in Z_p$，然后计算并输出 $F=h_1^f$ 和 $R_1=h_1^{r_f}$；

2) 主机随机选择 $r'\in Z_p$ 和 $r_2\in Z_p$，然后计算 $R'=h_2^{r'}$、$R_2=h_2^{r_2}$、$C=FR'$、$R=R_1R_2$ 和 $c_h=H_1(gpk\parallel C\parallel R)$，最后向 TCM 输入 c_h 和 n_I；

3) TCM 选择随机数 $n_T\in_R\{0,1\}^{2\lambda}$，计算 $c=H_1(c_h||n_I||n_T)$和 $s_f=r_f+cf(\mathrm{mod}\ p)$，最后将 $comm_T=(c,s_f,n_T)$返回给证明方主机；

4) 主机计算 $s_{r'}=r_2+cr'(\mathrm{mod}\ p)$，输出 $aux=F$ 以及 $comm=(C,c,s_f,s_{r'},n_T,n_I)$，并将 $comm$ 发送给凭证颁发方。

6.3.4 凭证颁发算法 2

该算法用于凭证颁发方为证明方生成匿名凭证，其输入、输出和算法流程如下：

a) 输入：证明方生成的凭证申请 $comm=(C,c,s_f,s_{r'},n_T,n_I)$、凭证颁发方的秘密信息 $isk=r$ 和公共参数 gpk。

b) 输出：(部分)匿名凭证(A,x,r'')。

c) 算法流程：凭证颁发方首先验证n_I的值是否是由自己生成且没有重放，随后凭证颁发方验证 $comm$ 是否有效，验证方式为：计算$R'=h_1^{s_f}h_2^{s_{r'}}C^{-c}$和$c_h'=H_1(gpk,C,R')$，验证 $c\overset{?}{=}H_2(c_h',n_I,n_T)$。然后随机选择$r''$，$x\in_R Z_p$，计算 $A=(g_1Ch_2^{r''})^{1/(x+r)}$，将$(A,x,r'')$发送给证明方主机。

6.3.5 凭证颁发算法 3

该算法用于证明方存储匿名凭证，其输入、输出和算法流程如下：

a) 输入：凭证颁发方为证明方生成的(部分)匿名凭证 $cre=(A,x,r'')$、验证所需信息 aux 以及公共参数 gpk。

b) 输出：若匿名凭证有效，则输出“有效”；否则输出“无效”。

c) 算法流程：证明方计算 $r=r'+r''(\mathrm{mod}\ p)$，验证 $e(A,wg_2^x)=e(g_1Fh_2^r,g_2)$是否成立。如果成立，将 $cre=(A,x,r)$可信存储。并输出“有效”，否则输出“无效”。

6.3.6 证明算法

该算法用于证明方进行匿名证明，生成匿名证明所需的信息，其输入、输出和算法流程如下：

a) 输入：凭证颁发方为证明方生成的匿名凭证 $cre=(A,x,r)$、验证所需信息 aux、待签名的消息 m(内含验证方提供的挑战随机数)、验证方的基本名 bsn 和公共参数 gpk。

b) 输出：证明信息 $\sigma=(B,K,T,c,s_f,s_x,s_a,s_b,n_T)$。

c) 算法流程：[下述步骤 1)～5)可以在算法调用前预计算]

1) TCM 随机选择$r_f\in_R Z_p$，计算并输出 $R=h_1^{r_f}$；

2) 主机选择 $a\in_R Z_p$，计算 $T=Ah_2^a$；

3) 主机计算 $b=ax+r(\mathrm{mod}\ p)$，选择$(r_x,r_a,r_b)\in_R Z_p\times Z_p\times Z$，计算$\widetilde{R}=T^{-r_x}h_2^{r_b}$，$\hat{R}=T_w^{r_a}$，$R_2=e(\widetilde{R}R,g_2)\hat{R}$；

4) 如果 $bsn=\perp$，主机随机选择 $d\in_R Z_p$并计算 $B=h_1{}^d$，$K=F^d$，$R_1=R^d$；如果 $bsn\neq\perp$，主机计算 $B=e(h_1,H_3(bsn))$，$K=e(F,H_3(bsn))$，$R_1=e(R,H_3(bsn))$；

5) 主机计算$c_h=H_1(gpk,B,K,T,R_1,R_2)$，当步骤 1)由 TCM 预计算时；

6) 主机向 TCM 输入(c_h,m,bsn)，TCM 随机选择$n_T\in_R\{0,1\}^{2\lambda}$，计算 $c=H_4(c_h,m,bsn,n_T)$以及$s_f=r_f+cf$，并输出(c,s_f,n_T)；

7) 主机计算$s_x=r_x+cx(\mathrm{mod}p)$，$s_a=r_a+ca(\mathrm{mod}p)$，$s_b=r_b+cb(\mathrm{mod}p)$，并输出证明信息 $\sigma=(B,K,T,c,s_f,s_x,s_a,s_b,n_T)$。

6.3.7 验证算法

该算法用于验证方验证证明方的匿名证明信息，其输入、输出和算法流程如下：

a) 输入：公共参数 gpk、签名的消息 m、验证方的基本名 bsn、证明方生成的匿名证明信息 $\sigma=(B,K,T,c,s_f,s_x,s_a,s_b,n_T)$。

b) 输出：若证明方的匿名证明信息验证通过，则输出“有效”；否则输出“无效”。

c) 算法流程：

1) 验证方首先验证s_f,s_x,s_a,s_b是Z_p上的元素，然后验证对于撤消列表上所有的值f_i，$K\neq B^{f_i}$；

2) 若 $bsn\neq\perp$且 $B\neq e(h_1,H_3(bsn))$，输出“无效”；

3) 验证方首先计算$R_2'=e(T,g_2^{-s_x}w^{-c})T_1^{c}T_2^{s_f}T_3^{s_b}T_w^{s_a}$，$R_1'=B^{s_f}K^{-c}$，再计算$c_h'=H_1(gpk,B,K,T,R_1',R_2')$和$c'=H_4(c_h',m,bsn,n_T)$，如果 $c\neq c'$输出“无效”，否则输出“有效”。

7 直接匿名证明接口

7.1 概述

TCM 安全芯片应提供 TCM_ECDAA_Setup、TCM_ECDAA_Join 和 TCM_ECDAA_Sign 三个匿

名证明命令支持，分别用于 ECDAA 证明的凭证颁发过程和匿名证明过程。由于 TCM_ECDAA_Join 和 TCM_ECDAA_Sign 这两个命令的运行将消耗 TCM 内部的多数资源，因而厂商可选择在它们执行期间禁用其他命令。

对于通用的 TCM 芯片，不同于普通的 TCM 命令，在同一次凭证颁发会话中的 TCM_ECDAA_Join 或同一次证明-验证会话中的 TCM_ECDAA_Sign 将分为多个阶段执行。每个阶段完成一个原子操作，各原子操作合作才能构成完整的直接匿名证明功能。由于每个原子操作只进行少量的运算和存储，多阶段方式一方面可以最大限度地降低 TCM 的资源需求，另一方面可以缩短匿名证明过程独占 TCM 的时间，减小对于用户使用 TCM 的影响。鉴于原子操作之间中断有可能使得 TCM 内部存储的公共参数发生变化，所以应在各执行阶段中检查公共参数的一致性。

TCM 芯片匿名证明实现上可以选择支持多个并行的匿名证明会话或仅支持单个匿名证明会话。如果仅支持单个匿名证明会话，则新调用的 TCM_ECDAA_Setup 命令将清除 TCM 内部原有的匿名证明数据结构。TCM 必须验证公共参数的合法性。公共参数必须由合法的凭证颁发方签名，否则敌手可能选择特殊的公共参数来与 TCM 执行 ECDAA 协议，进而获取 TCM 的秘密值 f 的部分信息。

本章规范了 TCM 匿名证明接口以及主要的内部运行流程，除下文规定的步骤之外，还应进行其他必要的运算和检查。例如，应检查通用参数（tag、paramSize、ordinal）和授权相关参数，详见表 1、表 2、表 4、表 5、表 7、表 8，并在发现错误的情况下按照普通 TCM 命令的相关规则予以处理。具体参数格式按 GM/T 0012 的规定执行。

7.2 TCM_ECDAA_Setup 命令

7.2.1 接口输入参数定义

接口输入参数定义 TCM_ECDAA_Setup 命令只能由 TCM 所有者向安全芯片 TCM 发起，命令执行前，TCM 所有者需先执行 TCM 的授权会话功能，TCM 分配相应授权会话句柄和序列号并返回给 TCM 所有者，然后 TCM 所有者才能按照表 1 所规范的命令格式执行 TCM_ECDAA_Setup 命令。表 1 规范了 TCM_ECDAA_Setup 命令接口的输入参数。

表 1 TCM_ECDAA_Setup 命令接口输入参数

序号	长度/字节	类型	名称	说明
1	2	TCM_TAG	Tag	命令类型标识
2	4	UINT32	paramSize	输入参数的总长度（字节数）
3	4	TCM_COMMAND_CODE	ordinal	命令码
4	4	TCM_HANDLE	handle	匿名证明会话句柄
5	1	BYTE	Stage	命令执行阶段
6	4	UINT32	inputSize0	inputData0 的长度（字节数）
7	<>	BYTE[]	inputData0	输入参数 0
8	4	UINT32	inputSize1	inputData1 的长度（字节数）
9	<>	BYTE[]	inputData1	输入参数 1
10	4	TCM_AUTHHANDLE	authHandle	属主授权会话句柄
11	<>	TCM_AUTHDATA	ownerAuth	主要输入参数的消息认证码，以属主授权值为密钥

注 1：长度值为“<>”意为参数长度与实现所采用的密码学算法有关。

注 2：输入参数中的消息认证码计算方法为：

HMAC(authData, HASH(ordinal||stage||inputSize0||inputData0||inputSize1||inputData1)||序列号)。

7.2.2 接口输出参数定义

表 2 规范了 TCM_ECDAA_Setup 命令接口的输出参数。

表 2 TCM_ECDAA_Setup 命令接口输出参数

序号	长度/字节	类型	名称	说明
1	2	TCM_TAG	Tag	命令类型标识
2	4	UINT32	paramSize	输入参数的总长度(字节数)
3	4	TCM_RESULT	returnCode	执行结果
4	4	UINT32	outputSize	输出数据长度
5	<>	BYTE[]	outputData	输出数据
6	<>	TCM_AUTHDATA	resAuth	主要输出参数的消息认证码,以属主授权值为密钥
注 1:长度值为“<>”意为参数长度与实现所采用的密码学算法有关。 **注 2**:输入参数中的消息认证码计算方法为: HMAC(authData, HASH(returnCode||ordinal||outputSize||outputData)||序列号),其中 ordinal 与输入参数相同。				

7.2.3 命令处理流程

7.2.3.1 流程概述

TCM_ECDAA_Setup 命令执行分为若干个阶段,表 3 规范了各阶段功能的详细定义。阶段 0 的接口输入句柄参数为空,输出参数为新的 Setup 会话句柄,作为阶段 1 和阶段 2 的接口输入句柄参数。阶段 1 为验证凭证颁发方公钥证书链操作,需要重复执行,重复次数为公钥证书链长度。

表 3 TCM_ECDAA_Setup 处理流程

阶段	输入参数 0 (InputData0)	输入参数 1 (inputData1)	操作	输出参数 (OutputData)	暂存数据
0	凭证颁发方公钥 证书链长度	无	初始化	Setup 会话 句柄	无
1	公钥	对公钥的签名	验证凭证颁发方 公钥证书链	无	公钥
2	凭证颁发方 公共参数	对公共参数的 签名值	设置公共参数	无	无

7.2.3.2 详细流程

7.2.3.2.1 阶段 0

命令处理流程如下:

a) 检查当前 TCM 中是否有足够资源执行 TCM_ECDAA_Setup。如果不足,则返回 TCM_RESOURCES(会话资源不足,需要释放已经建立的授权会话或者传输会话)或 TCM_NOSPACE(密钥存储空间不足,需要释放已经加载的密钥)。

b) 设置 ECDAA_issuerSettings 中的所有域为 NULL。

c) 设置 ECDAA_TCMSpecific 中的所有域为 NULL。
d) 设置 ECDAA_session 中的所有域为 NULL。
e) 验证输入参数 0 的长度不超过相应数据结构可存储数值长度的上限值，即验证 sizeOf(inputData0) == sizeOf(ECDAA_TCMSpecific -> ECDAA_count)，如不相等，则返回错误码 TCM_ECDAA_INPUT_DATA0。
f) 验证 inputData0 > 0，如不满足，则返回错误码 TCM_ECDAA_INPUT_DATA0。
g) 设置 ECDAA_TCMSpecific -> ECDAA_count = inputData0。
h) 为本次 Setup 会话分配会话句柄，并将 outputData 设置为该句柄。
i) 设置 ECDAA_session -> ECDAA_digestContext= Hash(ECDAA_TCMSpecific)。
j) 设置 ECDAA_session -> ECDAA_stage = 1。
k) 返回 TCM_SUCCESS。

7.2.3.2.2 阶段 1

命令处理流程如下：

a) 验证 ECDAA_session ->ECDAA_stage== 1，如不相等，则返回错误码 TCM_ECDAA_STAGE，清除会话句柄。
b) 验证 ECDAA_session -> ECDAA_digestContext== HASH(ECDAA_TCMSpecific)，如不相等，则返回错误码 TCM_ECDAA_TCM_SETTINGS。
c) 如果 ECDAA_session -> ECDAA_scratch == NULL：
   1) 设置 ECDAA_session -> ECDAA_scratch = inputData0。
   2) 设置 ECDAA_issuerSettings ->ECDAA_digest_k0 = HASH(ECDAA_session -> ECDAA_scratch)。

否则如果 ECDAA_session -> ECDAA_scratch ! = NULL：

   1) 设置 signedData = inputData0。
   2) 设置 signatureValue = inputData1。
   3) 以 ECDAA_session -> ECDAA_scratch 作为公钥，验证 signatureValue 是否为对 signedData 的合法签名，如果不是，则返回错误码 TCM_ECDAA_ISSUER_VALIDITY。
   4) 设置 ECDAA_session -> ECDAA_scratch = signedData。

d) ECDAA_TCMSpecific -> ECDAA_count 减 1。
e) 如果 ECDAA_TCMSpecific -> ECDAA_count ==0:ECDAA_session -> ECDAA_stage 加 1。
f) 设置 ECDAA_TCMSpecific -> ECDAA_digestIssuer= HASH(ECDAA_issuerSettings)。
g) 设置 ECDAA_session -> ECDAA_digestContext= HASH(ECDAA_TCMSpecific)。
h) 设置 outputData = NULL。
i) 返回 TCM_SUCCESS。

7.2.3.2.3 阶段 2

命令处理流程如下：

a) 验证 ECDAA_session ->ECDAA_stage==2，如不相等，则返回错误码 TCM_ECDAA_STAGE，清除会话句柄。
b) 验证 ECDAA_TCMSpecific -> ECDAA_digestIssuer == HASH(ECDAA_issuerSettings)，如不相等，则返回错误码 TCM_ECDAA_ISSUER_SETTINGS。
c) 验证 ECDAA_session -> ECDAA_digestContext== HASH(ECDAA_TCMSpecific)，如不相等，则返回错误码 TCM_ECDAA_TCM_SETTINGS。

d) 验证输入参数 0 的长度不超过相应数据结构可存储数值长度的上限值，即验证 sizeOf(inputData0) == sizeOf(TCM_ECDAA_ISSUER)，如不相等，则返回错误码 TCM_ECDAA_INPUT_DATA0。

e) 设置 X = ECDAA_issuerSettings ->ECDAA_digest_k0。

f) 设置 ECDAA_issuerSettings = inputData0。

g) 验证 X == ECDAA_issuerSettings ->ECDAA_digest_k0。如不满足，返回错误码 TCM_ECDAA_INPUT_DATA0。

h) 设置 signatureValue = inputData1。

i) 设置 signedData = ECDAA_issuerSettings。

j) 以 ECDAA_session -> ECDAA_scratch 为公钥，验证 signatureValue 是 signedData 的合法签名，如不满足，返回错误码 TCM_ECDAA_ISSUER_VALIDITY。

k) 设置 ECDAA_TCMSpecific -> ECDAA_digestIssuer = Hash(ECDAA_issuerSettings)。

l) 设置 ECDAA_session -> ECDAA_digestContext= HASH(ECDAA_TCMSpecific)。

m) 设置 ECDAA_session -> ECDAA_scratch = NULL。

n) 设置 outputData = NULL。

o) 返回 TCM_SUCCESS。

7.3 TCM_ECDAA_Join 命令

7.3.1 接口输入参数定义

接口输入参数定义 TCM_ECDAA_Join 命令只能由 TCM 所有者向安全芯片 TCM 发起，命令执行前，TCM 所有者需先执行 TCM 的授权会话功能，TCM 分配相应授权会话句柄和序列号并返回给 TCM 所有者，然后 TCM 所有者才能按照表 4 所规范的命令格式执行 TCM_ECDAA_Join 命令。表 4 规范了 TCM_ECDAA_Join 命令接口的输入参数。

表 4 TCM_ECDAA_Join 命令接口输入参数

序号	长度/字节	类型	名称	说明
1	2	TCM_TAG	Tag	命令类型标识
2	4	UINT32	paramSize	输入参数的总长度(字节数)
3	4	TCM_COMMAND_CODE	Ordinal	命令码
4	4	TCM_HANDLE	Handle	匿名证明会话句柄
5	1	BYTE	Stage	命令执行阶段
6	4	UINT32	inputSize0	inputData0 的长度(字节数)
7	<>	BYTE[]	inputData0	输入参数 0
8	4	UINT32	inputSize1	inputData1 的长度(字节数)
9	<>	BYTE[]	inputData1	输入参数 1
10	4	TCM_AUTHHANDLE	authHandle	属主授权会话句柄
11	<>	TCM_AUTHDATA	ownerAuth	主要输入参数的消息认证码，以属主授权值为密钥

注 1：长度值为“<>”意为输入参数长度与实现所采用的密码学算法有关。

注 2：输入参数中的消息认证码计算方法为：

HMAC(authData，HASH(ordinal||stage||inputSize0||inputData0||inputSize1||inputData1)||序列号)。

7.3.2 接口输出参数定义

表 5 规范了 TCM_ECDAA_Join 命令接口的输出参数。

表 5 TCM_ECDAA_Join 命令接口输出参数

序号	长度/字节	类型	名称	说明
1	2	TCM_TAG	Tag	命令类型标识
2	4	UINT32	paramSize	输入参数的总长度(字节数)
3	4	TCM_RESULT	returnCode	执行结果
4	4	UINT32	outputSize0	输出数据 0 长度
5	<>	BYTE[]	outputData0	输出数据 0
6	4	UINT32	outputSize1	输出数据 1 长度
7	<>	BYTE[]	outputData1	输出数据 1
8	<>	TCM_AUTHDATA	resAuth	主要输出参数的消息认证码,以属主授权值为密钥

注 1:长度值为“<>”意为输入参数长度与实现所采用的密码学算法有关。

注 2:输入参数中的消息认证码计算方法为:

HMAC(authData,HASH(returnCode||ordinal||outputSize0||outputData0||outputSize1||outputData1)||序列号),其中 ordinal 与输入参数相同。

7.3.3 命令处理流程

7.3.3.1 流程概述

TCM_ECDAA_Join 命令执行分为若干个阶段,表 6 规范了各阶段功能的详细定义。阶段 0 的接口输入句柄参数为 Setup 命令会话句柄,输出参数包含新的 Join 会话句柄,代替原 Setup 会话句柄,用于阶段 1 和阶段 2 的接口输入句柄参数。阶段 2 结束会清除该会话句柄。

表 6 TCM_ECDAA_Join 处理流程

阶段	输入参数 (inputData0)	输入参数 1 (inputData1)	操作	输出参数 0 (outputData0)	输出参数 1 (outputData1)	暂存数据
0	凭证颁发方公共参数	h_1, p	初始化公共参数 生成秘密信息 f $F=h_1^f$ 生成 r_f $R_1=h_1^{r_f}$	Join 会话句柄	F, R_1	p
1	$c_h=H_1(gpk\|\|C\|\|R)$	n_I	生成 n_T $c=H_2(c_h\|\|n_I\|\|n_T)$ $s_f=r_f+cf(\mathrm{mod}\ p)$	n_T	c, s_f	无
2	无	无	enc(ECDAA_TCMSpecific)	enc(ECDAA_TCMSpecific)	无	无

7.3.3.2 详细流程

7.3.3.2.1 阶段 0

命令处理流程如下：

a) 检查当前 TCM 中是否有足够资源执行 ECDAA_Join。如果不足，则返回 TCM_RESOURCES(会话资源不足，需要释放已经建立的授权会话或者传输会话)或 TCM_NOSPACE(密钥存储空间不足，需要释放已经加载的密钥)。
b) 设置 ECDAA_issuerSettings = inputData0。
c) 验证 ECDAA_issuerSettings 中的所有域都存在，否则返回错误码 TCM_ECDAA_INPUT_DATA0。
d) 设置 ECDAA_session 中的所有域为 NULL。
e) 为会话分配新句柄，设置 outputData0 为新句柄。
f) 验证 ECDAA_TCMSpecific -> ECDAA_digestIssuer == HASH(ECDAA_issuerSettings)，如不相等，则返回错误码 TCM_ECDAA_ISSUER_SETTINGS。
g) 设置 ECDAA_generic_h1 = inputData1 -> data0。
h) 验证 HASH(ECDAA_generic_h1) == ECDAA_issuerSettings -> ECDAA_digest_h1，如不相等，则返回错误码 TCM_ECDAA_INPUT_DATA1。
i) 设置 ECDAA_generic_p = inputData1 -> data1。
j) 验证 HASH(ECDAA_generic_p) == ECDAA_issuerSettings -> ECDAA_digest_p，如不相等，则返回错误码 TCM_ECDAA_INPUT_DATA1。
k) 利用随机数发生器 RNG 获取 32 字节长随机数，并将其存储于 X。
l) 设置 ECDAA_TCMSpecific -> ECDAA_rekey = X mod ECDAA_generic_p。
m) 设置 X = ECDAA_generic_h1，设置 Y = ECDAA_TCMSpecific -> ECDAA_rekey。
n) 计算 ECDAA_session -> ECDAA_scratch = X^Y。
o) 设置 outputData1 -> dataSize0 = sizeOf(ECDAA_session -> ECDAA_scratch)，outputData1 -> data0 = ECDAA_session -> ECDAA_scratch。
p) 利用随机数发生器 RNG 获取 32 字节长随机数，并将其存储于 X。
q) 设置 ECDAA_session -> ECDAA_rf = X mod ECDAA_generic_p。
r) 设置 X = ECDAA_generic_h1，设置 Y = ECDAA_session -> ECDAA_rf。
s) 计算 ECDAA_session -> ECDAA_scratch = X^Y。
t) 设置 outputData1 -> dataSize1 = sizeOf(ECDAA_session -> ECDAA_scratch)，outputData1 -> data1 = ECDAA_session -> ECDAA_scratch。
u) 设置 ECDAA_TCMSpecific -> ECDAA_digestIssuer = Hash(ECDAA_issuerSettings)。
v) 设置 ECDAA_session -> ECDAA_digestContext = HASH(ECDAA_TCMSpecific)。
w) 设置 ECDAA_session -> ECDAA_scratch = ECDAA_generic_p。
x) 设置 ECDAA_session -> ECDAA_stage = 1。
y) 返回 TCM_SUCCESS。

7.3.3.2.2 阶段 1

命令处理流程如下：

a) 验证 ECDAA_session -> ECDAA_stage == 1，如不相等，则返回错误码 TCM_ECDAA_STAGE，清除会话句柄。
b) 验证 ECDAA_TCMSpecific -> ECDAA_digestIssuer == HASH(ECDAA_issuerSettings)，

如不相等，则返回错误码 TCM_ECDAA_ISSUER_SETTINGS。

c) 验证 ECDAA_session -> ECDAA_digestContext == HASH(ECDAA_TCMSpecific)，如不相等，则返回错误码 TCM_ECDAA_TCM_SETTINGS。

d) 设置 ECDAA_generic_p = ECDAA_session -> ECDAA_scratch。

e) 设置 X = inputData0。

f) 设置 Y = inputData1。

g) 利用随机数发生器 RNG 获取 32 字节长随机数，并将其存储于 Z。

h) 设置 outputData0 = Z。

i) 设置 ECDAA_session -> ECDAA_scratch = $H_2(X||Y||Z)$。

j) 设置 outputData1 -> dataSize0 = sizeOf(ECDAA_session -> ECDAA_scratch)，outputData1 -> data0 = ECDAA_session -> ECDAA_scratch。

k) 设置 X = ECDAA_TCMSpecifc -> ECDAA_rekey。

l) 设置 Y = ECDAA_session->ECDAA_rf。

m) 设置 Z = ECDAA_session -> ECDAA_scratch。

n) 计算 ECDAA_session -> ECDAA_scratch = Y+(Z * X) mod ECDAA_generic_p。

o) 设置 outputData1 -> dataSize1=sizeOf(ECDAA_session -> ECDAA_scratch)，outputData1 -> data1 = ECDAA_session -> ECDAA_scratch。

p) 设置 ECDAA_TCMSpecific -> ECDAA_digestIssuer = Hash(ECDAA_issuerSettings)。

q) 设置 ECDAA_session -> ECDAA_digestContext= HASH(ECDAA_TCMSpecific)。

r) ECDAA_session -> ECDAA_stage 加 1。

s) 设置 ECDAA_session -> ECDAA_scratch = NULL。

t) 返回 TCM_SUCCESS。

7.3.3.2.3 阶段 2

命令处理流程如下：

a) 验证 ECDAA_session ->ECDAA_stage == 2，如不相等，返回错误码 TCM_ECDAA_STAGE，清除会话句柄。

b) 验证 ECDAA_TCMSpecific -> ECDAA_digestIssuer == HASH(ECDAA_issuerSettings)，如果相等，返回错误码 TCM_ECDAA_ISSUER_SETTINGS。

c) 验证 ECDAA_session -> ECDAA_digestContext== HASH(ECDAA_TCMSpecific)，如不相等，返回错误码 TCM_ECDAA_TCM_SETTINGS。

d) 计算 enc(ECDAA_TCMSpecific) 并填充数据结构 TCM_ECDAA_BLOB，设置 outputData0 为加密后的 TCM_ECDAA_BLOB 类型数据，加密密钥为 TCM 的 ECDAA 存储保护密钥 TCM_PERMANENT_DATA ->ecdaaBlobKey。

e) 设置 ECDAA_session -> ECDAA_scratch = NULL。

f) 设置 outputData1 = NULL。

g) 清除会话句柄。

h) 返回 TCM_SUCCESS。

7.4 TCM_ECDAA_Sign 命令

7.4.1 接口输入参数定义

TCM_ECDAA_Sign 命令只能由 TCM 所有者向安全芯片 TCM 发起，命令执行前，TCM 所有者需先执行 TCM 的授权会话功能，TCM 分配相应授权会话句柄和序列号并返回给 TCM 所有者，然后 TCM 所有者才能按照表 7 所规范的命令格式执行 TCM_ECDAA_Sign 命令。表 7 规范了 TCM_EC-

DAA_Sign 命令接口的输入参数。

表 7　TCM_ECDAA_Sign 接口输入参数

序号	长度(字节)	类型	名称	描述
1	2	TCM_TAG	tag	标识符
2	4	UINT32	paramSize	输入参数总长度(字节数)
3	4	TCM_COMMAND_CODE	ordinal	命令码
4	4	TCM_HANDLE	handle	匿名证明会话句柄
5	1	BYTE	stage	命令执行阶段
6	4	UINT32	inputSize0	输入参数 0 的长度(字节数)
7	<>	BYTE[]	inputData0	输入参数 0
8	4	UINT32	inputSize1	输入参数 1 的长度(字节数)
9	<>	BYTE[]	inputData1	输入参数 1
10	4	TCM_AUTHHANDLE	authHandle	属主授权会话句柄
11	<>	TCM_AUTHDATA	ownerAuth	主要输入参数的消息认证码,以属主授权值为密钥
注 1：长度值为“<>”意为输入参数长度与实现所采用的密码学算法有关。 **注 2**：输入参数中的消息认证码计算方法为： HMAC(authData，HASH(ordinal||stage||inputSize0||inputData0||inputSize1||inputData1)||序列号)。				

7.4.2　接口输出参数定义

表 8 规范了 TCM_ECDAA_Sign 命令接口的输出参数。

表 8　TCM_ECDAA_Sign 接口输出参数

序号	长度(字节)	类型	名称	描述
1	2	TCM_TAG	Tag	标识符
2	4	UINT32	paramSize	输入参数的总长度(字节数)
3	4	TCM_RESULT	returnCode	执行结果
4	4	UINT32	outputSize0	输出数据 0 的长度
5	<>	BYTE[]	outputData0	输出数据 0
6	4	UINT32	outputSize1	输出数据 1 的长度
7	<>	BYTE[]	outputData1	输出数据 1
8	<>	TCM_AUTHDATA	resAuth	主要输出参数的消息认证码,以属主授权值为密钥
注 1：长度值为“<>”意为输入参数长度与实现所采用的密码学算法有关。 **注 2**：输入参数中的消息认证码计算方法为： HMAC(authData,HASH(returnCode||ordinal||outputSize0||outputData0||outputSize1||outputData1)||序列号),其中 ordinal 与输入参数相同。				

7.4.3　命令处理流程

7.4.3.1　流程概述

TCM_ECDAA_Sign 命令执行分为若干个阶段,表 9 规范了各阶段功能的详细定义。阶段 0 的接

口输入句柄参数为空，输出参数包含新的 Sign 会话句柄，作为阶段 1 和阶段 2 的接口输入句柄参数。阶段 2 结束会清除该会话句柄。

表 9　TCM_ECDAA_Sign 接口处理流程

阶段	输入参数 0 (inputData0)	输入参数 1 (inputData1)	操作	输出参数 0 (outputData0)	输出参数 1 (outputData1)	暂存数据
0	凭证颁发方 公共参数	加密的 TCM 秘密信息	初始化公共参数 载入 TCM 秘密	Sign 会话 句柄	无	无
1	p	h_1	生成 r_f $R=h_1^{r_f}$	R	无	p
2	$\bar{c}=H_1(c_h\|\|bsn)$	m	生成 n_T $c=H_4(\bar{c}\|\|m\|\|n_T)$ $s_f=r_f+cf(\mathrm{mod}\ p)$	n_T	c、s_f	无

7.4.3.2　详细流程

7.4.3.2.1　阶段 0

命令处理流程如下：

a) 检查当前 TCM 中是否有足够资源执行 ECDAA_Sign。如果不足，则返回 TCM_RESOURCES(会话资源不足，需要释放已经建立的授权会话或者传输会话)或 TCM_NOSPACE(密钥存储空间不足，需要释放已经加载的密钥)。

b) 设置 ECDAA_issuerSettings = inputData0。

c) 验证 ECDAA_issuerSettings 中的所有域都存在，否则返回错误码 TCM_ECDAA_INPUT_DATA0。

d) 设置 ECDAA_session 中的所有域为 NULL。

e) 为本次 Sign 会话分配新句柄，设置 outputData0 为新句柄。

f) 设置 ECDAA_TCMSpecific = unwrap(inputData1)，解密密钥为 TCM_PERMANENT_DATA->ecdaaBlobKey。

g) 验证 ECDAA_TCMSpecific -> ECDAA_digestIssuer == HASH(ECDAA_issuerSettings)，如不相等，则返回错误码 TCM_ECDAA_ISSUER_SETTINGS。

h) 设置 outputData1 = NULL。

i) 设置 ECDAA_session -> ECDAA_digestContext = HASH(ECDAA_TCMSpecific)。

j) 设置 ECDAA_session -> ECDAA_stage = 1。

k) 返回 TCM_SUCCESS。

7.4.3.2.2　阶段 1

命令处理流程如下：

a) 验证 ECDAA_session -> ECDAA_stage==1，如不相等，则返回错误码 TCM_ECDAA_STAGE，清除会话句柄。

b) 验证 ECDAA_TCMSpecific -> ECDAA_digestIssuer == HASH(ECDAA_issuerSettings)，如不相等，则返回错误码 TCM_ECDAA_ISSUER_SETTINGS。

c) 验证 ECDAA_session -> ECDAA_digestContext == HASH(ECDAA_TCMSpecific)，如不相等，则返回错误码 TCM_ECDAA_TCM_SETTINGS。

d) 设置 ECDAA_generic_p =inputData0。
e) 验证 HASH(ECDAA_generic_p) == ECDAA_issuerSettings -> ECDAA_digest_p,如不相等,则返回错误码 TCM_ECDAA_INPUT_DATA0。
f) 设置 ECDAA_generic_h1 = inputData1。
g) 验证 HASH(ECDAA_generic_h1) == ECDAA_issuerSettings -> ECDAA_digest_h1,如不相等,则返回错误码 TCM_ECDAA_INPUT_DATA1。
h) 利用随机数发生器 RNG 获取 32 字节长随机数,并将其存储于 Y。
i) 设置 ECDAA_session->ECDAA_rf = Y mod ECDAA_generic_p。
j) 设置 Y = ECDAA_session->ECDAA_rf。
k) 设置 X =ECDAA_generic_h1。
l) 计算 Z = X^Y。
m) 设置 outputData0= Z。
n) 设置 outputData1 = NULL。
o) ECDAA_session -> ECDAA_stage 加 1。
p) 设置 ECDAA_session -> ECDAA_scratch = ECDAA_generic_p。
q) 返回 TCM_SUCCESS。

7.4.3.2.3 阶段 2

命令处理流程如下:

a) 验证 ECDAA_session -> ECDAA_stage==2,如不相等,返回错误码 TCM_ECDAA_STAGE,清除会话句柄。
b) 验证 ECDAA_TCMSpecific -> ECDAA_digestIssuer == HASH(ECDAA_issuerSettings),如不相等,返回错误码 TCM_ECDAA_ISSUER_SETTINGS。
c) 验证 ECDAA_session -> ECDAA_digestContext == HASH(ECDAA_TCMSpecific),如不相等,则返回错误码 TCM_ECDAA_TCM_SETTINGS。
d) 设置 ECDAA_generic_p = ECDAA_session -> ECDAA_scratch。
e) 设置 X = inputData0。
f) 设置 Y = inputData1。
g) 利用随机数发生器 RNG 获取 32 字节长随机数,并将其存储于 Z。
h) 设置 outputData0 =Z。
i) 设置 ECDAA_session -> ECDAA_scratch = H_4(X||Y||Z)。
j) 设置 outputData1 -> dataSize0 = sizeOf(ECDAA_session -> ECDAA_scratch),outputData1 -> data0 = ECDAA_session -> ECDAA_scratch。
k) 设置 X = ECDAA_TCMSpecifc->ECDAA_rekey。
l) 设置 Y = ECDAA_session->ECDAA_rf。
m) 设置 Z = ECDAA_session -> ECDAA_scratch。
n) 计算 ECDAA_session -> ECDAA_scratch = Y+(Z * X) mod ECDAA_generic-p。
o) 设置 outputData1 -> dataSize1 = sizeOf(ECDAA_session -> ECDAA_scratch),outputData1 -> data1 = ECDAA_session -> ECDAA_scratch。
p) 设置 ECDAA_session -> ECDAA_scratch = NULL。
q) 清除会话句柄。
r) 返回 TCM_SUCCESS。

附 录 A
（规范性）
直接匿名证明接口数据结构

A.1 基本数据类型定义

基本数据类型定义见表 A.1。

表 A.1 基本数据类型定义

类型	描述
BYTE	无符号字符类型
UINT16	无符号 16 位整型
UINT32	无符号 32 位整型

A.2 导出数据类型定义

导出数据类型定义见表 A.2。

表 A.2 导出数据类型定义

类型	定义	描述
TCM_COMMAND_CODE	UINT32	命令码类型
TCM_HANDLE	UINT32	句柄类型
TCM_AUTHHANDLE	TCM_HANDLE	授权句柄类型
TCM_TAG	UINT16	命令标识类型
TCM_STRUCTURE_TAG	UINT16	数据结构标识类型
TCM_NONCE	BYTE[]	随机数类型
TCM_DIGEST	BYTE[]	密码杂凑类型
TCM_AUTHDATA	BYTE[]	授权数据类型
TCM_RESULT	UINT32	执行结果类型

A.3 数据结构定义

A.3.1 TCM_ECDAA_ISSUER

TCM_ECDAA_ISSUER 用于表示 DAA Issuer 的公钥。

```
typedef struct tdTCM_ECDAA_ISSUER {
    TCM_STRUCTURE_TAG   tag;
    TCM_DIGEST   ECDAA_digest_p;
    TCM_DIGEST   ECDAA_digest_h1;
    TCM_DIGEST   ECDAA_digest_k0;
} TCM_ECDAA_ISSUER
```

TCM_ECDAA_ISSUER 数据结构见表 A.3。

表 A.3 TCM_ECDAA_ISSUER 数据类型

类型	成员域名称	描述
TCM_STRUCTURE_TAG	Tag	数据结构标识
TCM_DIGEST	ECDAA_digest_p	凭证颁发方公钥中 p 的摘要值
TCM_DIGEST	ECDAA_digest_h1	凭证颁发方公钥中 h1 的摘要值
TCM_DIGEST	ECDAA_digest_k0	凭证颁发方公钥中 k0 的摘要值

A.3.2 TCM_ECDAA_TCM

TCM_ECDAA_TCM 用于存储 DAA 过程中与 TCM 相关的参数信息。当该结构只能以密文形式被从 TCM 导出，且只能被创建该结构的 TCM 导入。

```
typedef struct tdTCM_ECDAA_TCM {
    TCM_STRUCTURE_TAG   tag;
    TCM_DIGEST   ECDAA_digestIssuer;
    BYTE[32]     ECDAA_rekey;
    UINT32       ECDAA_count;
} TCM_ECDAA_TCM
```

TCM_ECDAA_TCM 数据类型见表 A.4。

表 A.4 TCM_ECDAA_TCM 数据类型

类型	成员域名称	描述
TCM_STRUCTURE_TAG	Tag	数据类型标识
TCM_DIGEST	ECDAA_digestIssuer	TCM_ECDAA_ISSUER 结构的摘要值
BYTE[32]	ECDAA_rekey	ECDAA 过程中 TCM 产生的秘密信息 f
UNIT32	ECDAA_count	ECDAA 过程中凭证颁发方证书链长度

A.3.3 TCM_ECDAA_CONTEXT

TCM_ECDAA_CONTEXT 用于存储 ECDAA 过程的上下文信息。

```
typedef struct tdTCM_ECDAA_CONTEXT {
    TCM_STRUCTURE_TAG   tag;
    TCM_DIGEST   ECDAA_digestContext;
    BYTE[]   ECDAA_scratch;
    BYTE[]   ECDAA_rf;
    BYTE    ECDAA_stage;
} TCM_ECDAA_CONTEXT
```

TCM_ECDAA_CONTEXT 数据类型见表 A.5。

表 A.5　TCM_ECDAA_CONTEXT 数据类型

类型	成员域名称	描述
TCM_STRUCTURE_TAG	Tag	数据类型标识
TCM_DIGEST	ECDAA_digestContext	上下文信息的摘要
BYTE[]	ECDAA_scratch	用于存储 ECDAA 过程中各阶段间传递的参数信息
BYTE[]	ECDAA_rf	TCM 产生的用于隐藏 f 的随机数
BYTE	ECDAA_stage	用于记录当前 ECDAA 进行到的阶段信息

A.3.4　TCM_PERMANENT_DATA

TCM_PERMANENT_DATA 给出了针对 ECDAA 过程 TCM_PERMANENT_DATA 中需要增加的变量。

```
typedef struct tdTCM_PERMANENT_DATA{
    TCM_KEY    ecdaaBlobKey;
}TCM_PERMANENT_DATA
```

TCM_PERMANENT_DATA 数据结构见表 A.6。

表 A.6　TCM_PERMANENT_DATA 数据类型

类型	成员域名称	描述
TCM_KEY	ecdaaBlobKey	ECDAA 存储保护密钥

A.3.5　TCM_STANY_DATA

TCM_STANY_DATA 给出了针对 ECDAA 过程 TCM_STANY_DATA 中需要增加的变量。

```
typedef struct tdTCM_STANY_DATA{
    TCM_ECDAA_ISSUER      ECDAA_issuerSettings;
    TCM_ECDAA_TCM         ECDAA_TCMSpecific;
    TCM_ECDAA_CONTEXT     ECDAA_session;
}TCM_STANY_DATA
```

TCM_STANY_DATA 数据类型见表 A.7。

表 A.7　TCM_STANY_DATA 数据类型

类型	成员域名称	描述
TCM_ECDAA_ISSUER	ECDAA_issuerSettings	数据颁发方信息
TCM_ECDAA_TCM	ECDAA_TCMSpecific	TCM 内部信息
TCM_ECDAA_CONTEXT	ECDAA_session	证明会话信息

A.3.6　TCM_ECDAA_BLOB

TCM_ECDAA_BLOB 用来表示 JOIN 过程后得到的数据块。

```
typedef struct tdTCM_ECDAA_BLOB {
```

```
    TCM_STRUCTURE_TAG          tag;
    BYTE[16]                       label;
    TCM_DIGEST                 blobIntegrity;
    UINT32                         additionalSize;
    [size_is(additionalSize)]      BYTE * additionalData;
    UINT32                         sensitiveSize;
    [size_is(sensitiveSize)]       BYTE * sensitiveData;
}TCM_ECDAA_BLOB
```

TCM_ECDAA_BLOB 数据类型见表 A.8。

表 A.8 TCM_ECDAA_BLOB 数据类型

类型	成员域名称	描述
TCM_STRUCTURE_TAG	Tag	数据类型标识
BYTE[16]	label	自定义标签
TCM_DIGEST	blobIntegrity	DAA_TCMSpecific 的摘要值
UINT32	additionalSize	additionalData 的长度
BYTE	additionalData	TCM 自定义数据
UINT32	sensitiveSize	sensitiveData 的长度
BYTE	sensitiveData	记录加密后的 DAA_TCMSpecific 内容

A.3.7 TCM_ECDAA_IOBLOB

TCM_ECDAA_IOBLOB 用来表示 SIGN 过程的输入输出数据块。

```
typedef struct tdTCM_ECDAA_IOBLOB {
    TCM_STRUCTURE_TAG          tag;
    UINT32                         dataSize0;
    [size_is(dataSize0)]           BYTE * data0;
    UINT32                         dataSize1;
    [size_is(dataSize1)]           BYTE * dataSize1;
}TCM_ECDAA_IOBLOB
```

TCM_ECDAA_IOBLOB 数据类型见表 A.9。

表 A.9 TCM_ECDAA_IOBLOB 数据类型

类型	成员域名称	描述
TCM_STRUCTURE_TAG	Tag	数据类型标识
UINT32	dataSize0	data0 的长度
BYTE	data0	data0 数据
UINT32	dataSize1	data1 的长度
BYTE	Data1	data1 数据

A.4 错误码定义

错误码定义见表 A.10。

表 A.10 错误码定义

序号	标识	解释
1	TCM_RESOURCES	资源不足
2	TCM_NOSPACE	存放密钥的空间不足
3	TCM_ECDAA_INPUT_DATA0	DAA 命令的输入参数 0 参数(一致性校验)有误
4	TCM_ECDAA_INPUT_DATA1	DAA 命令的输入参数 1 参数(一致性校验)有误
5	TCM_ECDAA_STAGE	DAA 命令的 stage 参数错误
6	TCM_ECDAA_ISSUER_SETTINGS	Issuer 的公钥信息一致性校验错误
7	TCM_ECDAA_TCM_SETTINGS	TCM 的内部信息一致性校验错误
8	TCM_ECDAA_ISSUER_VALIDITY	Issuer 的证书链校验错误

附 录 B
（资料性）
直接匿名证明椭圆曲线参数与辅助函数

B.1 椭圆曲线参数选择

给定一个椭圆曲线 E/F_q，一个基点 $G\in E(F_q)$，其阶为 n，一个点 $P\in E(F_q)$，则椭圆曲线离散对数问题是找到一个整数 $x\in[0,n-1]$，使得 $P=xG$。基于椭圆曲线的密码体制的安全性是建立在椭圆曲线离散对数问题难解的基础上的。对于椭圆曲线上点形成的加法群和该群中一个阶为 n 的基点 G 来说，目前比较有效的离散对数求解算法为 Pollard-rho 算法。

而对于易于计算对运算的椭圆曲线群来说，其嵌入度为 k，则通过计算对运算，Menezes-Okamoto-Vanstone 算法可以将椭圆曲线上点形成的加法群中的离散对数问题规约到F_{q^k} 上的离散对数问题，因此必须也要保证F_{q^k} 上的离散对数问题是难解的。对于F_{q^k} 上的离散对数问题，目前比较有效的算法是指数计算算法。

在保证了安全性的前提下，有限域F_q 应该尽可能的小，使得群上点运算的计算效率尽可能的高。F_{q^k} 也需要尽可能的小，这样可以提高对运算的效率。

在本文件中，推荐使用对运算友好的曲线，即嵌入度尽量小，提高对运算效率。G_1，G_2，G_T 安全强度相同。

B.2 辅助函数

TCM 需要计算两类特殊的杂凑值，其算法定义为：$H_2:\{0,1\}^{6\lambda}\rightarrow Z_p$ 和 $H_4:\{0,1\}^{*}\rightarrow Z_p$。这两类杂凑算法的输出限制为 Z_p 中的值。实现方式可以采用标准密码杂凑算法的输出模 p 为输出。

参 考 文 献

[1] ISO/IEC Trusted Computing Group TPM MainPart 1 Design Principles, Specification Version 1.2

[2] Trusted Computing Group TPM MainPart 2 TPM Structures, Specification version 1.2

[3] Trusted Computing Group TPM MainPart 3 Commands, Specification Version 1.2

[4] IEEE Std 1363—2000 Standard Specifications for Public-Key Cryptography

[5] ISO/IEC 15946 Information technology—Security techniques—Cryptographic techniques based on elliptic curves

[6] Chen X, Feng D. Direct anonymous attestation for next generation TPM[J]. Journal of Computers, 2008, 3(12): 43-50.

[7] Brickell E, Li J. A pairing-based DAA scheme further reducing TPM resources[C]. Trust and Trustworthy Computing. Springer BerlinHeidelberg, 2010: 181-195.

[8] Xi L, Qin Y, Feng D. Formal analysis of ECC-based Direct Anonymous Attestation schemes in Applied Pi Calculus[C]. Information Security Conference (ISC 2013). 2013.

[9] Xi L, Feng D. Formal Analysis of DAA-Related APIs in TPM 2.0[C]. Network and System Security (NSS 2014). Springer International Publishing, 2014: 421-434.

[10] Xi L, Feng D, Qin Y, et.al. Direct Anonymous Attestation in practice: Implementation and efficient revocation[C]. Privacy, Security and Trust (PST 2014). IEEE, 2014: 67-74.

[11] Xi L, Yang K, Zhang Z, et.al. DAA-Related APIs in TPM 2.0 Revisited[C]. Trust and Trustworthy Computing. Springer International Publishing, 2014: 1-18.

[12] 杨波，冯登国，秦宇等. 基于可信移动平台的直接匿名证明方案研究[J]. 计算机研究与发展，2014，51(7)：1436-1445.

[13] Zhang Q, Zhao S, Xi L, et.al. Mdaak: A flexible and efficient framework for direct anonymous attestation on mobile devices [C]. International Conference on Information and Communications Security. 2014.

[14] Yang B, Feng D, Qin Y. A Lightweight Anonymous M-shopping Scheme Based on DAA for Trusted Mobile Platform[C]. IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2014). IEEE, 2014: 9-17.

[15] Yang B, Yang K, Qin Y, et.al. DAA-TZ: An Efficient DAA Scheme for Mobile Devices using ARM TrustZone[C]. Trust and Trustworthy Computing. 2014.

中华人民共和国密码
行业标准
可信计算平台直接匿名证明规范
GM/T 0079—2020
*
中国标准出版社出版发行
北京市朝阳区和平里西街甲2号(100029)
北京市西城区三里河北街16号(100045)
网址 www.spc.net.cn
总编室:(010)68533533 发行中心:(010)51780238
读者服务部:(010)68523946
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 1.75 字数 53 千字
2021年5月第一版 2021年5月第一次印刷
*
书号:155066·2-35851 定价 36.00 元

GM/T 0079-2020

码上扫一扫 正版服务到