CPK认证系统

技术白皮书

北京大学 信息科学技术学院
网络与信息安全实验室
2004年6月

引言

随着以Internet及其所承载的网上业务的迅速发展，网络应用模型及其运行方式都发生了根本性的转变，网络由封闭的计算机网络发展为开放的互联网络，而网络业务则由简单的数据通信，发展成为网上电子交易，而在网络世界中如何保证交易的可信性、真实性的问题正变得越来越重要。交易安全是随电子交易的发展而提出的新概念，而实现交易安全的基本技术手段是认证系统。认证系统为电子交易（transaction）提供鉴别性证明、负责性证明和审计服务，即AAA安全构件：

- 鉴别性（Authentication），提供实体本身身份的证明。
- 负责性（Accountability），提供行为（Action）负责性证明和内容（Contents）负责性证明。
- 审计性（Audit），提供交易中的各种监管信息，包括回执证明。

与传统的保障数据安全的保密系统不同，认证系统是为电子商务和电子政务建立信任体系的证明系统，在交易安全中占有重要的位置。在认证系统中最核心的问题是密钥管理问题，因为密钥不仅用于密钥交换，也用于数字签名；不仅提供鉴别性服务，也提供负责性服务，也用来划定安全域，提供逻辑隔离技术手段。

在网络世界中，认证系统的密钥管理需要解决两个课题：一是密钥管理的规模化，二是基于标识的密钥分发。目前的解决方案有两种：一是依靠PKI技术构建的认证系统，二是依靠CPK技术构建的认证系统。两者各有其特点，也各有其适用的范围。

本技术白皮书详细介绍了CPK认证系统的特点、原理、体系结构、功能及典型应用案例，并与PKI进行了对比。

一、概述

1. 网络安全与交易安全

随着网络技术的迅速发展，网络由封闭的计算机网络发展为开放的互联网络，网络业务则由简单的数据通信，发展为网上电子交易。信息安全的内容也随之变化扩展。信息安全大致经历了三个发展过程：数据安全、系统安全、交易安全。

- 数据安全：是计算机网络时代的基本安全要求，主要涉及与数据安全相关联的技术手段，包括传输数据和存储数据的安全，最基本的安全构件为CIA，即Confidentiality（机密性），Integrity（完整性），Access Control（访问控制）。数据的机密性和完整性一般采用密码算法加以保障，而访问控制则需与授权机制结合。
- 系统安全：又称网络安全，是开放的互联网络时代的基本安全要求，主要涉及与信息系统运行相关的技术手段，包括边界和计算环境安全，最基础的安全构件是PDR：即Protection（保护），Detection（探测），Response（响应）。所采用的基本技术包括防火墙、VPN、IDS入侵检测、防病毒等。
- 交易安全：是网络电子交易时代的基本安全要求，为交易（或事务transaction）提供鉴

别性证明、负责性证明和审计服务。最基础的安全构件是SOA：即主体鉴别性（Subject Authentication）、客体鉴别性（Object Authentication）和行为鉴别性（Action Authentication）。

安全构件的分类不是绝对的，相同的安全构件在不同的环境中可以起到不同的作用，而相同的技术也可以构成不同的安全构件。例如，网上电子交易往往是以数据交换的方式进行，因此客体鉴别往往被数据安全所替代。同是数字签名，在数据安全中只对数据本身和传输行为负责；而在交易安全中则侧重于对内容负责。在电子银行业务中，一个票据可以构成一份数据，由发送方签名，对这份数据提供负责性。但如果数据中包括发行银行、银行行长、财务部门、银行帐号、法人代表等印章内容，发送方的签名显然不能对这些内容提供负责性，而这正是认证系统所要研究的对象。同样是密码技术，在系统安全中可用于VPN的加密隧道，以构建专用通道；在数据安全中可用于加密文件，以保护机密性；而在交易安全中则将其用于数字签名，以提供鉴别性和负责性服务。

2. 认证系统

在物理世界中，身份主要指一个人的社会地位和法律地位，行为主体往往指在法律意义上具有行为能力的人。因此身份认证、身份鉴别都是对人而言的，所用的技术也是照片、指纹等生物技术。人与人之间的区别，也主要体现在生物特征上。但在某些场合下采用生物特征来鉴别身份并不合适，会带来很大的不便。于是便产生了主体的逻辑替代物——标识，如姓名、称谓、个人身份号等。

在网络世界中身份的含义发生了很大变化。随着Agent等自动化软件技术的发展，越来越多的业务由进程、agent等代理人自动完成。此时，主体不再仅仅指人，进程也可以成为主体，以标识的名义进行动作，成为行为主体。

行为主体和行为对象构成了主客体关系，但这种关系式相对的，在一种情况下的主体，在另一种环境重可能成为客体。如：一个人管帐目，人是主体，帐目是客体。从一个帐号中作存取款动作，帐目成为主体，而存取款的金额变成客体。

在现实世界中，同一个行为主体可以有多个标识。人名、个人身份号、个人帐号、个人电话号码都可以作为行为主体的标识。实际上，在不同的业务系统中，标识的作用是不同的，如在人事档案管理中，人名和职务占有重要地位，在金融系统中，帐号占有重要的地位，在电子邮件通信中，邮件地址占有重要地位，而在电话业务中，电话号码则占有重要地位，等等。在网络世界认证系统中，通常通过标识的识别达到身份的识别的目的，在大多数情况下，身份和标识不会引起混淆，但是需要将两者加以区别，并注意两种情况：一是确定标识的识别和鉴别是否足以证明身份识别和鉴别；二是在标识的识别和鉴别就能够满足业务需求的情况下，不应强行扩展出身份真伪的结论。

作为信息安全领域的新概念，交易安全需要借助于证系统来保证。认证系统的核心问题是密钥管理，而密钥管理则需要解决两个难题：一是密钥的规模化管理，二是基于标识的密钥分发。目前，可用的认证系统主要有两类实现方式，一是采用PKI技术实现，另一种则采组合公钥技术（Combined Public Key，简称CPK）实现。CPK从技术体制本身解决了密钥管理规模化和密钥本身即可证明标识的课题，不需要可信第三方的证明，具有系统简捷高效，可行性好的优点。

在信息系统中，主体间认证指的是一个主体（验证方）对另一主体（证明方）进行识别（identify），验明身份（proof of identity）的过程。为完成认证，证明方必须具有某种“信物”，如：合同，协议，照片，证件，ID卡或身份证等，这种“信物”或证件的特征应事先在验证方处记录在案，并且主体和证件应具有一体性。因此，在认证系统中，主体认证必须满足注

册性、一体性和管辖性的要求。

- **注册性**

  在认证系统中，证明方用于认证的特征应预先设置或约定，即事先注册。由于认证通常是采用对照的方式实现的，因此没有事先注册的参照物就没有判别的依据，也就无法实现认证。注册中应包含认证所需的所有相关内容。注册性决定了一个验证主体管辖的证明方是有边界的，并明确定义了认证系统的安全域及管辖性。注册方和被注册方形成验证方/与证明方的主、从关系。如果在证明方与验证方之间存在主从关系，则可进行直接级认证。如果在证明方与验证方之间不存在直接的主从关系，则需要借助于第三方认证进行推理认证。但认证系统的第三方应是主管方，如在银行系统中，并且推理层次不应超过一级（单层KMC机制），以保证所需信任度。

- **一体性**

  在验证方进行验证时，证明方必须提供持证人和证件的一体性证明，如照片或人脑中的口令等。在进行当面验证时，可以提供生物特征（指纹、虹膜、照片等），也可以提供逻辑特征（密钥、随机数等参数）；在不能进行当面验证的环境中，则应提供可验证的逻辑参数，这种认证一般需要借助于相应的认证协议来实现。

- **管辖性**

  在认证系统中，交易双方在完成注册之后便归属于同一安全域，即属于统一管辖。在多层结构的认证链系统中，其管辖性存在差别，这种层次结构所导致的差别会影响认证系统的信任关系。

  此外，认证系统在实现过程中还应满足以下几种要求：

- 在签名机制上，支持多种签名。不同类型的签名间相互独立，签名生存期与对象的生存期一致。
- 在密钥交换机制上，对密钥加密密钥（二级密钥）应采用物理分发和静态管理。密钥的作用域，可分专用和通用。专用密钥只在定义的专用范围内起作用。同时设置级别控制密钥，实现不同级别的加密，以配合对存储数据的访问控制。
- 在层次结构上，能够支持不依托第三方的单层集中式管理模式，即由一个密钥管理中心统一管理的单层认证系统，以保证认证过程具有较高的信任度。

随着电子交易与认证系统的发展，密码技术的性质也随之发生变化。密码技术有许多不同用途，如在VPN中，密码被用于建立并维护安全的通信隧道，其目的除了对数据提供加密保护外，还可起到明确界定虚拟私有网络边界的作用。而在主体认证中，密码手段则只用作验证明对方的身份，其目的并不在数据的保密。由此可见，密码可根据其用途的划分为不同的类型：用于保证数据机密性的加密密码，和用于进行身份认证的认证密码。

在网络世界的交易活动中，密码技术的应用已逐渐普及到个人，而个人的私密和国家秘密是完全不同范畴的东西，因此认证密码与数据加密密码也归属于不同的范畴，不能简单地同一对待。所以，在电子交易中如何管理认证密码就成为一个新问题。认证理论认为，用于认证的密码算法必须公开。其原因在于，密码算法只有公开并得到公认，才能用于身份认证的目的，才能起到公断的作用。现代密码理论主张“一切秘密寄偶于密钥之中”，而密码算法则是可以公开的。最彻底实现这一理论的体制就是PKI认证体制。PKI体制完全脱离了以秘密手段达到保密目的思路，提出了以公开的手段达到保密目的的新思想。在PKI中，除了私钥一切都可以公开。

3. 密钥管理

在网络世界中，认证系统的作用是为交易安全提供可靠的鉴别性证明和负责性证明。

而一个合理、可用的认证系统则需要恰当的密钥管理技术和正确的技术路线的支撑。在大规模分布式网络环境中，密钥管理需要解决两个课题：一是密钥管理的规模化，二是基于标识的密钥分发。各国都围绕这两大课题相继开展了研究，取得了不同的进展和成果。

3.1 密钥管理的发展

自20世纪70年代中期开始，信息网络逐渐从点对点的通信网络发展为多点的交换网络，并由计算机网络发展为互联网络。交换网络的出现和发展，对密钥管理提出了许多新问题，其中最为突出的是：

1. 不依赖秘密信道的密钥分发问题；
2. 基于标识的密钥分发问题；
3. 密钥管理的规模化问题。

国外密钥管理技术的研究进展

1976年，Diffie-Hellman提出了著名的D-H密钥分发体制，首次解决了不依赖秘密信道的密钥分发问题，但这种体制只能用于会话密钥的交换，而且不能抵抗中间人攻击（attack in the middle）。公钥则以电话簿的形式予以公布。

1978年，Kohnfelder提出了Certificate Agency（CA认证机构）的概念，在CA集中式管理的模式下，采用密钥动态分发的管理体制，公钥以CA证书形式公布，私钥仍采用秘密（物理）信道分发，企图解决密钥的规模化问题。

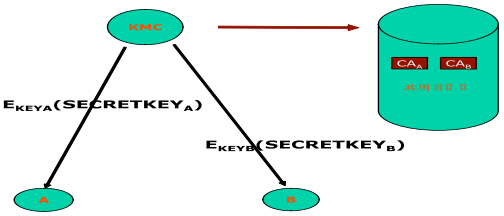


图1.1 Kohnfelder 密钥分发体系

1991年相继出现了PGP、PEM，首次提出密钥由个人生成的分散式密钥管理体制，以不传递私钥的方式避开了建立秘密信道的难题。各依赖方自行建立各自的密钥环，将常用的对方公钥存储在自身设备中。

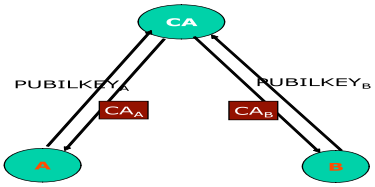


图1.2 密钥个人生成体系

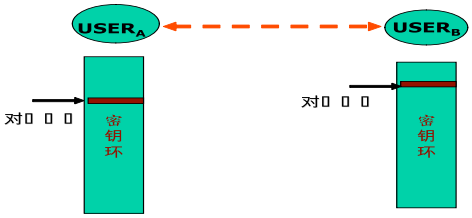


图1.3 PGP系统的密钥环

1996年，出现SPKI解决方案。PKI设立CA认证中心，证明公钥和标识的一体性，防止他人冒名顶替；通过构造多层的CA架构来解决密钥的规模化问题。PKI的另一进展是利用数字签名的功能，企图构建网上的认证体系，大大推动了认证理论的发展。

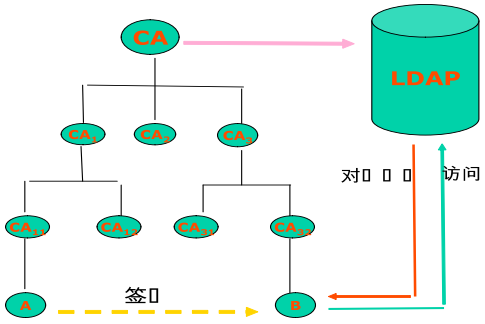


图1.4 PKI的公钥证书体制

我国密钥管理技术的研究进展

1989年，首次提出了双重密钥算法**SAD**(Symmetric and asymmetric doubled key)，成功解决了十万用户的密钥管理，并成功应用于某大型网络系统中。

1996年，提出了多重公钥算法**LPK** (Lapped public key)和密钥映射算法，并采用硬件芯片解决了100万用户的公钥管理和基于标识（身份）的密钥分发，用于我国第一个商用密码SJY01系统中。

2000年，提出了组合公钥算法**CPK** (combined public key)，用一个芯片解决了10^{48}个用户的公钥管理。CPK算法意义及应用前景正逐步被人们所认识，目前CPK在实际认证系统中应用的时机已经成熟。

2003年，提出了组合密钥算法**CSK** (combined symmetric key)，解决了海量对称密钥的分发问题。

在规模化密钥管理技术上，我国学者一直坚持集中式管理的模式，提出了基于密钥管理算法和密钥映射算法的新思路，在密钥管理研究上取得了突破性进展。显然，我国密钥管理技术所走的技术路线，与国外的技术路线不同，其特点是将复杂的密钥管理技术建立在一种算法基础上，企图以简便的方式解决基于标识的密钥分发和在单层KMC下密钥管理的规模化，以适应认证系统的需要。

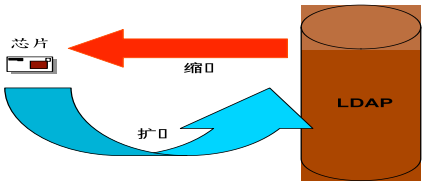


图1.5 缩小证书库的两种思路

与PKI相比，CPK体制解决问题的思路是简洁、直观的。如果应用系统中有一亿个用户，那么LDAP就需要保存一亿个证书，但能否将LDAP缩小到能够装进一个芯片中？这个想法不可行。那么反过来，是否能将一个较小集合扩大为一亿个证书呢？这个思路是可行的，其关键是寻求实现这一思路的技术途径，这就是所谓种子化的密钥或称组合密钥。一旦能将一亿个公钥装在一个芯片中，将对密钥管理产生巨大影响，密钥管理将变得非常简单，可以为构建合理的认证系统提供很好的技术保障。

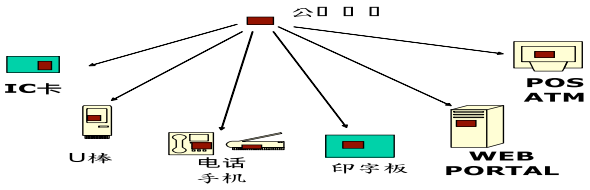


图1.6 芯片化证书库的应用

纵上可见，密钥管理发展的历史，可划分为三个阶段：

第一个阶段，1976－1978年间，公钥的发布从D-H的电话簿演化为Kohnfelder:的CA证书；

第二个阶段：1988:－1996年间，提出密钥个人生成的思路，公钥的发布从密钥环发展到LDAP，多层CA结构的创立；；

第三个阶段：1999－2003年间，提出种子化密钥技术，解决了基于标识的密钥和规模化密钥管理问题，为构建新型认证系统提供了很好的技术基础。

将三个阶段密钥管理的特点归结起来，比较如下：

表1.1 A密钥技术发展三阶段特点比较

	管理模式	证明方式	维护方式	证书属性
第一阶段	集中式	主管方	动态	公钥证书
第二阶段	分散式	第三方	动态	公钥证书
第三阶段	集中式	标识自身	静态	标识证书

在第三阶段，密钥管理技术有了突破性的进展，证书不再是公钥证书，而是成为了分发私钥的工具，变成了ID证书。公钥可以公布，公钥本身证明或代表其标识，没有被顶替的危险，因而用不着第三方的公证。所以，密钥管理系统变得非常简洁，大大增强了建立新型认证系统的可行性和可靠性。

3.2 密钥管理体制

密钥管理的主要任务是准确定义各种密钥的作用域，支持密码技术的运用，并灵活地组成各种封闭环境和开放环境。封闭粒度可以是个人，通信双方、小范围、大范围等，因此，密钥管理是实现各种隔离技术的最佳技术手段。

密钥管理是加密系统（密钥交换等）和认证系统（数字签名等）的基础。不同网络拓扑和不同业务要求，都会对密钥管理提出不同需求。在密钥管理中，从需求出发，满足需求的最简单密钥管理方案即是最好的解决方案。例如，有些系统不需支持事后不可否认性，就没有必要设置数字签名，这样可以避免不必要的复杂性。

认证系统是建立交易信任的基础。认证系统的核心是合理的签名机制，而签名机制的实现则需要依赖合理的密钥管理。因此，缺乏恰当的密钥管理技术，势必难于较好地实现密钥交换机制、签名机制和认证机制。

现有密钥管理技术可划分为两大类：

- KMI：由密钥管理中心实现密钥的统一集中式管理的，但传统的KMI一般只能支持较小规模的密钥管理，而CPK做为一种新兴的KMI技术则较好地解决了大规模密钥的管理问题；
- PKI：依靠第三方证明的多层CA分散式密钥管理。

一般集中式密钥管理适用于各种专用网，而分散式密钥管理则适用于开放网。

密钥管理涉及密钥的生产、验证、分发、交换、存储、更换、销毁等多个方面，涉及密钥在其整个生存周期中的所有处理，另外还涉及密钥的行政管理制度和管理人员的素质。KMI，特别是CPK与PKI相比在密钥管理方面有着自身的特点：

（1） 密钥生产体制

在KMI的集中式密钥管理模式下，密钥分发依靠物理通道或秘密通道进行。密钥变量可以保密，包括私钥和公钥。交易中的安全责任由密钥管理中心承担，因此，普遍为官方所采用。在PKI的分散式管理模式下，公钥的发布不依赖秘密信道，但需经第三方公证。交易中的安全责任完全由个人承担。因此，适用于安全责任不损害第三方利益的场合，如当个人的失误不影响国家利益或他人利益时。分散机制还要求建立CA证书作废系统，并且且当提供加密功能时必须实现私钥的托管。

（2） 密钥维护体制

密钥维护有两种方法：静态维护和动态维护。

集中式管理采用静态维护的机制，几乎没有系统维护量，但密钥的更换则较不方便。密钥变量的相对固定却对数字签名机制有利，因为所保存文件的签名寿命应与文件寿命相同。

分散式管理采用动态维护的的机制，密钥更换非常方便，但密钥变量的更换，需要建立CA证书作废系统，以识别过期密钥和当前有效密钥，而且对存储文件的签名不利。在动态管理的模式中，目录库LPAD必须一直在线运行，其维护量很大，运行经费也很高。

（3） 密钥存放体制

密钥存放形式现有两种：分散保存和集中保存。

在集中式管理的模式下，将所有公钥存储在专用媒体中，一次性面对面发放给各用户分散保存和使用，其协议非常简单，又很安全。电脑黑客的入侵破坏，也只能破坏本机而不影响其他终端或用户。分散式管理的模式则采用公钥集中保存和使用，需要解决公钥不被冒名顶替的问题和公共目录库自身的安全。

（4） CA的层次结构

密钥管理结构有两种：单层和多层。

集中式管理多采用主管方的单层管理结构，这种结构可以建立用户和中心的直接级信任关系，在用户之间建立一级推理级信任。

分散式管理采用多层CA结构，这种结构只能建立顾客之间的二级以下的推理信任。层次越多，越淡化（dilute）信任。

（5） 运行体制与效率

证书运行方式有两种：不需要运行工具和依靠运行工具。

集中式ID证书可以不需要运行工具，公钥不需要传递，因为公钥已经分散存储在每一用户手中；也可以使用公用目录公布公钥，此时必须使用专门的运行工具（如PKI）；而分散式条件下，向通信对方索要公钥的方式获得公钥，或从共用目录中获得对方公钥，因此不

得不借助于专门的运行工具。而在实际环境中，这种运行工具（如PKI）部署复杂，而运行效率却不高。

3.3 密钥分类与设置

每一种密钥都有自己的作用域，如：用于密钥交换、用于数字签名，用于身份鉴别，或用于网络隔离等等。认证系统的特点就是各不同作用域的密钥分别起作用，从而最终实现对交易安全的保障。密钥按其作用域的不同可分为多种密钥类型，如用于认证的签名密钥，用于数据加密的通信密钥，用于身份鉴别的特征参数等等。

（1） 通信密钥

通信密钥是保证通信双方认证通信的主要参数变量，也是划分或定义作用域的主要逻辑手段。在现代通信中，通信的形式大都集中在一个地址到另一地址的通信（电子邮件）、一个序列号到另一序列号的通信（电话、设备）。因此，通信密钥主要由地址密钥和序列号密钥组成。通信密钥，也可以作为数字签名密钥，对消息源负责，对传输行为提供负责性证明。通信密钥按通信方式划分为共用密钥、对通密钥、指译密钥。

①共用密钥：用于通播通信，用对称密钥实现。可以在不同范围内用单密钥实现共用密钥（如在局域网、省域网、全国网中）。

②对通密钥：交信双方共有的密钥，用对称密钥实现，

③指译密钥：只有指定通信方才能脱密的密钥，用非对称密钥实现。

对通密钥和指译密钥是密钥管理的最基本技术。按密钥的作用域不同，认证网络划分为不同规模的网络等，均由对通密钥和指译密钥构成。不同的认证子网间的密钥互相独立，形成各自封闭的子网，子网之间的通信则由上一级的认证网络密钥负责。如果网络规模大于几万用户时，则应考虑采用PKI或CPK方式实现密钥管理。

（2） 签名密钥

在认证系统中，签名密钥主要用于对内容负责。例如，在银行交易中内容主要包括：票据序列号、帐号、法人姓名、单位名称。数字签名主要对序列号、帐号、法人、单位负责。

办公通信的内容主要是以信息的敏感性来划分的；金融交易的内容，则主要以金额来划分，例如，小额交易的负责性由帐号签名来负责，中额交易的负责性由帐号签名和法人签名负责，大额交易的负责性则有帐号、法人、单位签名来负责。

签名密钥的作用域可以与通信密钥的作用域相同，也可以不同。一般对签名的作用域的要求没有通信密钥那样严格。因为签名是印章系统，签名者签完名后任何人都可以验证。超出作用域的签名，由于不具有效力所以是否能验证也就无关紧要了。

（3） 分级密钥

在认证系统中，密钥还可用于划分各种级别；如划分为，绝密、机密、秘密、内部；或高级职员级、中层职员级、一般职员级、顾客级等；不同级别分配不同密钥，可以实现访问范围的控制。

但分级密钥的级别应支持向下兼容，如高级职员级享有高层职员、中层职员、一般职员、顾客级的处理能力；中层职员级享有中层职员、一般职员、顾客级处理能力；一般职员享有

一般职员、顾客级处理能力；顾客级则只有顾客级处理能力。

3.4 密钥管理中心（KMC）

在KMI体制中，密钥管理主要由密钥管理中心（KMC）集中实现，密钥由密钥管理中心统一编制，统一生产，统一配发使用。密钥管理中心具有以下特点与职能：

（1） 集中式管理

在KMI中，采用的是集中式的密钥管理体制，全网设立密钥管理中心（KMC），统一管理密钥的生成、分发、更换。对大规模分布式网络而言，可能需要在各地设立密钥管理分中心，在中心和分中心之间应建立秘密通道，分中心可以分担中心的密钥管理任务，如果各地的网络是独立的封闭网，则分中心可承担本网内部的密钥管理。

（2） 定义密钥的作用域

在认证系统中，密钥被用于认证、密钥交换和网络隔离。因此密钥管理中心的首要职能是定义各种密钥的作用域和划分粒度，明确管理的职责。

密钥是划分或定义作用域的主要逻辑手段。一种密钥只属于一种作用域，按着密钥的作用域，密钥可以划分为：小范围的作用域，如：限制在局域网内的密钥。中范围的作用域，如：限制在省域网内的密钥。大范围的作用域，如：限制在全国网内的密钥。

（3） 权限的授权

密钥管理中心还应承担一定的权限管理职能，负责人员授权级别划分，如划分为绝密、机密、秘密、内部等，用于实现访问控制，并确定分级控制的基本材料，如密级、范畴等。给不同级别的人员配置不同的密钥，还可实现不同级别的加密认证。

认证卡是实现分级控制、分级加密、身份鉴别的重要工具，也是密钥参数的重要工具。各种级别都是往下兼容的，高一级级别可以认证同级和下级级别，但低级级别不能认证高一级以上级别。

随着网络电子交易的发展，认证系统的ID证书越来越成为网上身份证明。由于网络上的认证与通信有着紧密联系，因此，ID证书要素应包括各种通信密钥、各种签名密钥以及其他与认证、通信相关的密钥和参数，如：系统审计的基本材料，如姓名、单位等个人资料；

二、组合公钥密码系统

在公开密钥技术出现以前，如何在大规模的网络环境中进行大规模密钥的分配和管理一直是一个难题。公开密钥技术的出现解决了这一难题，但CA在实现用户与其公钥的绑定的同时，CA本身也带来了新的技术和管理问题。

在一般的公钥体制中，各用户的公钥是直接公布的，有多少用户，就公布多少个公钥，而在组合公钥密码系统（CPK）中，各用户的公钥不直接公布，而只公布公钥因子矩阵，各用户的公钥则通过公钥因子矩阵和相关映射值计算出来。

1. CPK体系结构

组合公钥的体系结构主要由五个部分组成（见图1）：

- 密钥管理中心，负责初始化系统的公/私钥因子矩阵，发布公钥因子矩阵，接受名字管理中心的申请，产生指定终端实体（名）的私钥并将其返回给名字管理中心。
- 注册管理中心，接受终端实体的申请并代理终端实体向密钥管理中心申请私钥，管理、维护并发布统一的名字空间，存储和发布作废的终端实体名称。
- 终端实体，即为使用系统安全功能的用户、应用程序或设备。
- 管理实体，制定整个系统的安全实施策略并负责管理密钥管理中心的实体。
- 资料库，供公开访问的系统信息库，包括所有的公钥因子表、有效名字和作废名字信息。

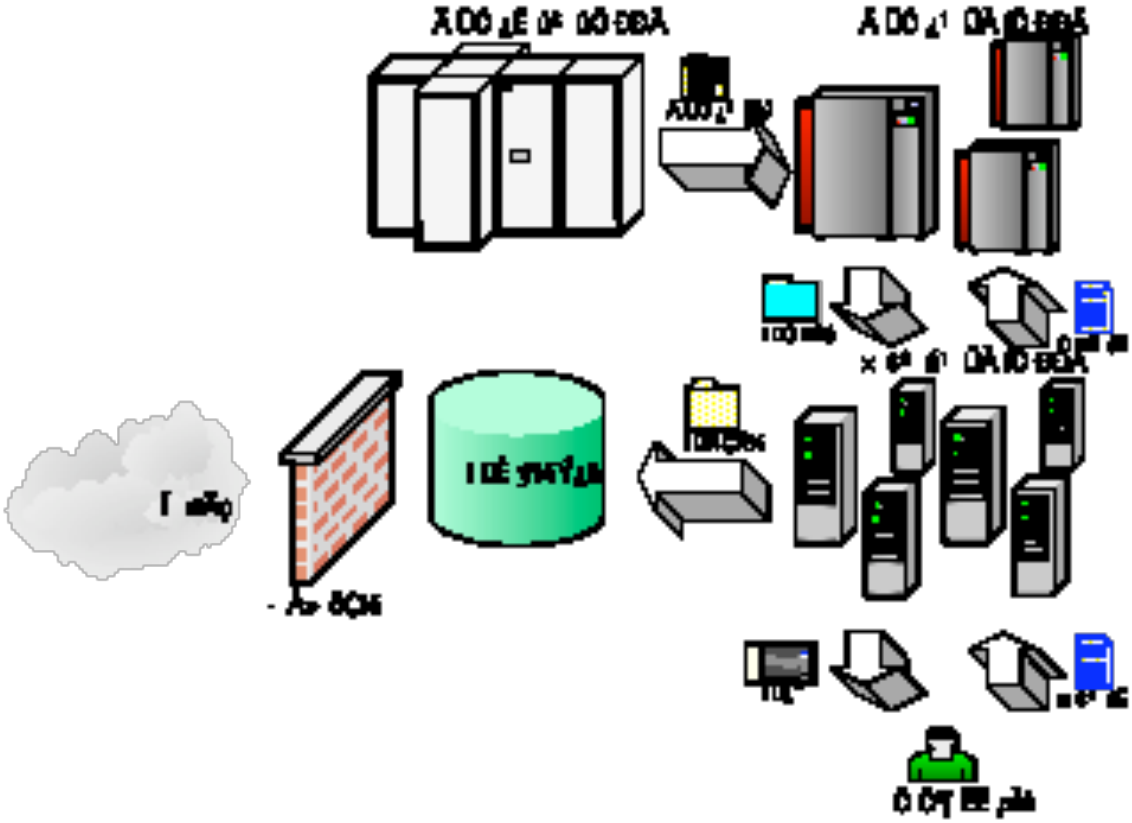


图2.1 组合公钥系统的体系结构

在采用组合公钥技术的系统中，私钥分发的基本流程分为以下五步：

1）终端实体/用户：向注册管理中心提出注册申请。

2）注册管理中心：检查用户名，防止不同用户使用相同的用户名，防止不同用户名享有相同的映射值；注册用户名，保留用户的所有曾用名(挂失)和在用名；然后通过安全信道向密钥中心提出证书申请。

3）密钥管理中心：根据用户名计算私钥，生成用户ID证书并对其签名，并通过安全信道将用户ID证书发回给注册管理中心。

4）注册管理中心：将用户私钥写入ID卡、CA卡或其它安全的媒体中，通过秘密信道（如加密信道、面对面、物理方式）分发给署名用户。如果公钥因子矩阵是存放在某种媒体中，那么就连同该媒体一起分发给用户，这就使用户能够一次性获得所有的公钥。

5）终端实体/用户：用自己的口令对私钥进行加密，以保证私钥的机密性。

2. CPK原理

本节将讨论基于素数域椭圆曲线密码系统的组合公钥技术的实现原理和机制。

假定椭圆曲线密码系统的参数如下：

$$T = \{a, b, G, N, p\}$$

其中，a、b为（1）式中的椭圆曲线的参数，G为椭圆曲线上的基点，N为椭圆曲线上点的个数，也即椭圆曲线的阶，p即为素数域F_p的阶。

如果假设用户的私钥为素数域F_p上的任一整数$SK = r$，那么对应的公钥PK即为椭圆曲线E上的点$r \cdot G$。

2.1 密钥因子矩阵

将一定数量的公、私钥因子对分别组成公钥因子矩阵和私钥因子矩阵。其中，矩阵的列表示组合的层次，行表示每层所包含的密钥变量。设矩阵大小为m行n列，那么，矩阵就分为n层，每层的密钥变量为m个。密钥组合因子矩阵由密钥管理中心离线产生。

设私钥因子矩阵和公钥因子矩阵的大小为$m \times n$（m行n列）。私钥因子矩阵SSK中的元素即为整数标量r_{ij}，而公钥因子矩阵中的元素即为与私钥因子r_{ij}对应的椭圆曲线上的点$r_{ij} \cdot G$。在密钥管理中心生成公、私钥因子矩阵后，私钥因子矩阵需要保持秘密，而公钥因子矩阵则予以公布。

私钥因子矩阵

$$SSK = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{21} & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}$$

公钥因子矩阵

$$PSK = \begin{bmatrix} r_{11} \cdot G & r_{12} \cdot G & \cdots & r_{1n} \cdot G \\ r_{21} \cdot G & r_{22} \cdot G & \cdots & r_{2n} \cdot G \\ \vdots & \vdots & \cdots & \vdots \\ r_{m1} \cdot G & r_{m2} \cdot G & \cdots & r_{mn} \cdot G \end{bmatrix}$$

2.2 映射算法

在组合公钥体系中，用户实体的公（私）钥是采用映射算法从公（私）钥因子矩阵中计算出来的。

最简单的映射算法是采用随机映射，根据用户名（设备名）计算出对应的映射值，从而将用户名（设备名）与对应的映射值联系起来。用户名可以是在专用网中使用的用户名，也可以是单位名称，个人身份证号，银行帐号或IP地址等，设备名可以是序列号，电话号码等，只要是验证方所认可的名称即可。在这种情况下，用户名一经确定，映射值也就跟着确定了。因此已公布的用户名不能轻易更改。如果需要更改，就需要将更改后的用户名及时通知验证方。显然，在此用户名和公钥是捆绑在一起的，这与CA和PKI的宗旨是相同的。

假设因子矩阵的大小为$m \times n$，因子矩阵分为n层，那么就需要n个映射值。由于每层的因子量均为m，则映射算法F_i（X）的n次映射为：

$$F_1（用户名）\bmod m = map_1;$$
$$F_2（用户名）\bmod m = map_2;$$
$$\cdots \qquad \cdots \qquad \cdots$$
$$F_n（用户名）\bmod m = map_n;$$

其中，函数F_i（X）可以是采用不同密钥K_i的加密或散列函数。

如，F_i（X）可以是 E_{K_i}（用户名）mod $m = map_i$，其中 E_{K_i}（X）为3DES或AES等加密算法。

2.3 密钥计算

在对用户名进行映射，获得n个映射值之后，就可以按组合的方式计算出用户的公、私钥。设映射层次为n，一个用户名（A）的n个映射值分别为（i_1，i_2，…，i_n）。根据这些映射值取出私钥因子矩阵SSK的第k（k = 1, 2, …, n）列的第i_k行个元素，即将映射值解释为第一列的i_1行，第二列的i_2行，…，第n列的i_n行，这样就可以从私钥因子矩阵SSK中取出n个因子（$r_{i_1 1}$，$r_{i_2 2}$，…，$r_{i_n n}$）。那么用户A的私钥即为：

$$SK_A = （r_{i_1 1} + r_{i_2 2} + \cdots + r_{i_n n}） \bmod m,$$

显然，私钥运算是模m下的n个整数的加法。由于私钥因子矩阵是组合公钥系统中的秘密矢量，所以只能在密钥管理中心（KMC）计算私钥，然后将用户A的私钥SK_A以物理方式秘密分发给用户A。

当验证方需要获得用户A的公钥时，由于用户名的映射算法对于一个具体的系统来说是唯一的，所以验证方同样可以根据A的用户名和映射算法计算出与用户A相对应的n个映射值（i_1，i_2，…，i_n）。同样将这些映射值看作是公钥因子矩阵第一列的i_1行，第二列的i_2行，…，第n列的i_n行，就可以从公钥因子矩阵PSK中取出与（$r_{i_1 1}$，$r_{i_2 2}$，…，$r_{i_n n}$）相对应的n个公钥因子（$r_{i_1 1} \cdot G$，$r_{i_2 2} \cdot G$，…，$r_{i_n n} \cdot G$）。由于公钥因子矩阵是公开的，所以任何验证方都可以计算出用户A的公钥：

$$PK_A = （r_{i_1 1} \cdot G + r_{i_2 2} \cdot G + \cdots + r_{i_n n} \cdot G），$$

在此，所采用的加法是椭圆曲线E上的点的加法，而进行n－1次点的加法的计算代价是很小的。

经过这样的计算，用户A保有的私钥SK_A和各验证方计算出的用户A的公钥PK_A就是一一对应的。有了公私钥对，就可以用基于椭圆曲线密码系统的数字签名(DSS)或D-H密钥交换协议等来实现数字签名和密钥交换。

3. 功能模块

CPK系统可以定义为由一系列硬件、软件、政策、流程以及人机界面综合组成的认证系统，其目的是实现ID证书的生成、发放、管理、更新以及作废。

CPK系统由功能相对独立的多个实体构成，实体之间通过通信协议进行通信，从而将整个系统构成一个有机的整体。实体可以是计算机系统，在此计算机系统实体包括硬件、软件和进行设置的人员，也可以是在系统中完成特定任务的人、软件或设备，例如使用ID证书的用户和设备是终端实体，而在系统中负责管理并制定策略的管理员是管理实体。

CPK体系结构由四个主要的功能模块构成：密钥生产中心、密钥管理中心、注册管理中心和资料库。此外，这些功能模块的运行还需要终端实体或管理实体的参与，并采用一定的数据结构、格式与协议实现相互之间的通信。

3.1 密钥生产中心KPC

密钥生产中心简称KPC（Key Production Center）。在CPK系统中，KPC的核心职能是负责生产、发放、备份并管理CPK系统初始化与运行所必须的系统参数。系统参数统一规定了CPK系统参数的选取，包括椭圆曲线算法和公私钥因子矩阵，在2.1节CPK原理中已对这些参数的作用进行了说明。在一个分布式的CPK系统中有且仅有一个唯一的密钥生产中心，但是可以由多个密钥管理中心分别负责不同的网络或安全域，密钥生产中心将针对各个密钥管理中心生成不同的系统参数表，密钥管理中心则根据自己的系统参数表进行初始化，并以此为基础生成ID证书。KPC由三个子功能模块构成，分别用于生产KPC公私钥对、KPC公钥证书和系统参数表，3.1.2节详细描述三个子模块的功能、接口和实现方式。

3.1.1 KPC的部署

由于KPC的职能是负责生产、存储、备份CPK系统的系统参数，包括系统的核心机密信息——密钥因子矩阵，因此对要求有较高的要求，并且只在密钥管理中心初始化或全面更新的时候才执行其系统参数生产工作，因此KPC可以离线运行，无需联网。KPC必须部署于安全计算机系统并且与网络物理隔离，只有最高权限的管理人员才能够在物理上访问密钥生产中心，设定相关参数并生成CPK系统参数表，并由管理人员通过物理信道将其发到密钥管理中心。

3.1.2 系统参数表的生成

系统参数表规定了CPK系统中一部分重要的参数，包括采用的椭圆曲线和密钥因子矩阵。这些参数以系统参数表数据类型进行明确的定义。生成系统参数表是KPC的主要作用。

3.1.2.1 系统参数表的定义

本节采用标准的ASN.1描述系统参数表的数据结构。ASN.1是一种数据定义语言，广泛用于描述网络协议和与Internet标准相关的数据类型，对ASN.1的详细说明可以参见资料[还没找到]。以后的各章节还会采用ASN.1语言来描述其他数据结构。这里只是简要的介绍一下ASN.1 语法规则。

- 在ASN.1声明中，变量名称写在它们的类型前面，这与大部分编程语言不同。
- 变量名称总以小写字母开头；数据类型总以大写字母开头。
- ASN.1的关键字全部由大写字母构成，如SEQUENCE、CHOICE、ENUMERATED。
- 变量可以通过ASN.1赋值符号“::=”进行定义。

系统参数表的ASN.1定义如下：

```
SystemParameterTable ::= SEQUENCE {
    version          Version DEFAULT v1,          -- CPK系统的版本号
    SystemSerial     IMPLICIT UniqueIdentifer     -- 系统参数唯一编号
    keyLength        KeyLength,                   -- ECC私钥长度

    ellipticCurve    EllipticCurve,               -- 椭圆曲线参数，详细定义见3.5节
    keyMatrix        KeyMatrix,                   -- CPK的公、私钥矩阵
```

```
    issueTime           Time,                   -- 参数的发布时间
    issuer              Issuer,                 -- 参数的发布者
    signatureAlgorithm  SignatureAlgorithm,     -- 所用的签名算法
    signatureValue      OCTET STRING            -- 对参数表的签名
}

KeyMatrix ::= SEQUENCE {
    matrixID          IMPLICIT UniqueIdentifer,     -- CPK公、私钥矩阵的唯一编号
    matrixRowNum          INTEGER,                  -- 矩阵的行数
    matrixColumeNum       INTEGER,                  -- 矩阵的列数
    publicMatrix          SEQUENCE OF ECPoint,      -- 公钥因子矩阵
    protectedPrivateMatrix    ProtectedPrivateMatrix,   -- 私钥因子矩阵
}

ProtectedPrivateMatrix ::= SEQUNCE {
    encryptionAlgorithm       EncryptionAlgorithm,  -- 保护私钥因子的加密算法
    encryptedKey              OCTET STRING,         -- 加密私钥（用口令加密）
    encryptedMatrix           OCTET STRING          -- 加密的私钥因子矩阵
}
```

系统参数表中各参数的数据类型的具体定义参见3.5节ID证书定义中的同名数据类型。

3.1.2.2 椭圆曲线参数的生成算法

椭圆曲线密码算法的原理可以参见2.1，该算法的具体描述如下：

输入：密钥长度keyLength
输出：ECParameters（a, b, G, N, p）格式的椭圆曲线参数

执行：

1) 将keyLength转化为整数t，t = keyLenth×16，即当keyLength为枚举值key-112-Bits时，t为112；key为key-160-Bits时，t为160。

2) 选择一个素数p，$[\log_2 p] = t$，p决定了曲线所在的有限域$\mathbb{F}_q$；

选择BigInteger类型的整数a，b（BigInteger ::= BIT STRING）。a，$b \in \mathbb{F}_q$，是椭圆曲线E:

$y^2 = x^3 + ax + b \pmod p$的系数。

3) 选择曲线E: $y^2 = x^3 + ax + b \pmod p$ 上的一个基点$G(x, y)$，计算G的幂N，N是一个素数。计算余因子$h = \#E(\mathbb{F}_q)/N$。要保证：

- $4a^3 + 27b^2 \neq 0 \pmod p$

- $\#E(\mathbb{F}_p) \neq p$
- $p^B \neq 1 (\mathrm{mod}\, N)\ ,\ 1 \leq B \leq 20$
- $h \leq 4$

最后输出ECParameters(a, b, G, N, p)

3.1.2.3 密钥矩阵的生成算法

密钥因子矩阵的参数m、n表示矩阵的行长度和列长度，具体的值由相应的输入参数决定，应该保证m、n的值在区间[16，256]之内。密钥矩阵由私钥因子矩阵privateMatrix和公钥因子矩阵publicMatrix构成。每个矩阵都含有$m \times n$个因子。不同的是，privateMatrix中的元素是大整数BigInteger，而publicMatrix中的元素是椭圆曲线点ECPoint。两种矩阵中的元素都是按照线性排列表示的，排列顺序都是行优先的。如图所示，逻辑上的三行四列矩阵在privateMatrix和publicMatrix中都是线性表示的。

11	12	13	14
21	22	23	24
31	32	33	34

11	12	13	14	21	22	23	24	31	32	33	34

privateMatrix中的每个元素都是一个随机生成的BigInteger，大小要保证在区间[1，N-1]之内。其中N是椭圆曲线参数之一（ECParameters.n）。

publicMatrix中的每个元素都是一个椭圆曲线点ECPoint，这个点通过privateMatrix矩阵中相同位置上的大整数和椭圆曲线参数ECParameters.G相乘得到。

3.1.2.4 数据的编码标准

CPK中的基本数据类型包括：

- ENUMERATED
- BIT STRING
- GeneralizedTime
- OCTET STRING
- INTEGER

而基本的结构则包括：

- SEQUENCE
- CHOICE
- OPTIONAL

基本数据类型及结构的具体定义如下，

- ENUMERATED：保证枚举类型不超过256，用一个字节进行编码。
- BIT STRING：保证标志数量不超过 8，用一个字节进行编码，在i386体系结构上由最低端的位表示第一个标志，高端的位表示随后的标志。
- GeneralizedTime：编码为 1 4 个数字的字节序列，分别为YYYYMMDDHHMMSS，用来表示格林威治时间的年月日时分秒，其中年有 4 个字符表示，小时为 2 4 进制。例如，

20041012183000。

- OCTET STRING：如果字节序列的长度小于２５５，编码中的第一个字节为长度，其余字节为字节序列的值。否则，编码的前两个字节或四个字节表示长度，后面表示值。长度的编码依照INTEGER的编码方式进行。具体表示长度的数据为1、2还是4，由编码解码的程序负责判断，不在编码中进行说明。
- INTEGER：由固定长度的字节序列表示，但是编码中不包含字节序列的长度信息。编码中第一个字节是大整数的最低位字节，以此类推。
- SEQUNCE：SEQUNCE中包含的多个子元素按顺序进行编码，而其作为整体的SEQUENCE信息不进行编码表示。
- CHOICE：编码中的第一个字节表示CHOICE中第几个子元素进行了编码，后面的字节序列为该子元素的编码。
- OPTIONAL：如果一个数据类型有OPTIONAL标志，那么在对该属性编码前，首先增加1byte的标志用于表示该属性是否存在，如果标志的值为零则该属性不存在，后面的数据是属于下一个属性。否则按照正常的方式在后面添加该可选属性的编码数据。

3.2 密钥管理中心KMC

密钥管理中心（KMC-Key Management Center）的主要职能是在系统中提供生成证书的服务。

针对不同的应用环境，密钥管理中心可以有两种ID证书生成方式：

- 自动批量成产方式

在一些应用中，终端实体的标识是由KMC按照一定规则自动生成的，这些规则保证了生成的标识在整个系统中是唯一的。例如在信用卡应用中，信用卡的号码由银行指定而不是由用户指定，银行方面只要保证信用卡号码不重复即可。CPK应用于信用卡业务，采用信用卡号码作为标识，可以采用自动批量生产方式。KMC可以预先生成大量的证书，并发布到注册管理中心，其中的标识是由KMC指定的。

- 请求应答方式

CPK系统中终端实体的标识往往需要终端实体来指定。例如在邮件的安全应用中，终端实体在加入CPK系统之前已经拥有邮件帐号，并需要将这个邮件帐号作为自己在CPK系统中的标识。因为无法预知终端实体的邮件帐号，KMC必须在接收到RMC的证书申请之后才能根据确定终端实体的标识是什么，并根据这个标识来生成证书。请求-应答方式的使用比较广泛。如果CPK系统使用一些已有的标识，如电子邮件帐号，电话号码、身份证号码、用户的真实姓名等，都需要使用请求-应答方式。

3.2.1 KMC的部署

生成证书方式的不同，KMC的部署也不同。在请求-应答方式中，KMC随时可能接收到RMC的证书生成请求，因此KMC一定要和RMC联网并随时在线。KMC应该部署在安全的内部网之中，并通过防火墙进行保护。而在批量自动生产方式中，KMC不需要来自RMC的请求就可以一次性地生成大量的证书，这些证书可以预先保存到RMC，RMC在需要时可以将其写入认证卡并分发给终端实体。因此在批量自动生产方式中，KMC除了部署在内部网上，也完全可以部署在和网络隔绝的计算机上，在批量生成证书后，由管理员通过移动存储器物理的分发到RMC。

3.2.2 KMC的数据

KMC本地保存的数据包括：

- 系统参数表，需要加密保存
- KMC生成的全部证书文件

3.2.3 KMC的接口

安装KPC公钥证书

输入：KPC公钥证书。由KMC管理员在KMC本地指定输入文件。
输出：操作是否成功
执行：

- 验证KPC公钥证书格式是否完整，如果格式不正确返回“失败”
- 通过口令加密的方式将KPC公钥证书加密存储在本地磁盘，返回“成功”

安装系统参数表

输入：系统参数表。由KMC管理员在KMC本地指定输入文件。
输出：操作是否成功
执行：

- 通过管理员口令解密KPC公钥证书
- 读取KPC公钥证书中的KPC公钥
- 验证系统参数表的KPC签名是否正确，如果不正确返回“失败”
- 验证系统参数表的格式是否正确，如果不正确返回“失败”
- 将系统参数表加密存储在本地，并附加散列值用于检查是否篡改

生成对公开参数表

公开参数表用于RMC和资料库的初始化。公开参数表和系统参数表非常相似，只是比系统参数表缺少了私钥因子矩阵，因为私钥因子矩阵只能存储在KMC本地。

公开参数表的ASN.1描述[to do]

输入：系统参数表。由KMC管理员在KMC本地指定加密的系统参数表文件。
输出：公开参数表。
执行：

- 解密系统参数表
- 验证系统参数表的有效性，如果已经被篡改，返回“失败”并报告该严重系统错误。
- 读取系统参数表中数据，并写入对应的公开参数表中。
- 将公开参数表写入文件，作为输出。

生成证书

由于证书格式还没有最终确立，本节还无法细化

证书申请表的格式：todo
证书的格式：参见2.3.2.3证书的格式一节

输入：证书申请表
输出：证书
执行：
生成证书

此处加入KMC和RMC具体的通信协议格式，还需要参考一下已有的一些标准是怎么定义通信格式的。是采用类似TCP的二进制头还是采用HTTP的ASCII头。

3.3 注册管理中心RMC

在CPK系统中，注册管理中心（RMC Register Management Center）直接面向终端实体，向终端实体提供证书申请、证书发放和证书作废等服务，因此注册管理中心是CPK系统和用户之间的接口。

注册管理中心通常以机构的形式存在，需要一定数量的管理人员。例如在企业中应用CPK系统，注册管理中心需要企业人事部门的参与，来进行注册审核等工作的管理。注册管理中心由三个部门构成，注册部、生产部和发放部。下面将详述每个部门的功能与实现。

3.3.1 注册部

注册部是RMC中直接面向终端实体的部分。注册部接收终端实体的请求，和其他功能模块合作进行注册申请、证书更新、证书作废的工作。这些工作都需要对终端实体的身份和权限进行审核，因此通常需要企业的人事管理部门的参与。

（1） 证书的申请：终端实体必须首先进行注册才能获得认证卡，进行安全应用。证书的申请的详细过程参见4.1证书的申请一节。

（2） 证书的更新：终端实体向注册管理中心申请重新获得一张认证卡。证书的更新过程和证书的申请过程类似，由于终端实体在证书的申请过程中已经进行过注册，更新过程可以参照申请过程进行相应的简化其中的注册过程。

（3） 证书的作废：证书的作废主要是对证书中标识的作废。终端实体向注册管理中心提出作废证书的申请，注册管理中心会在系统范围内作废证书中的标识，这样就不会再生产出具有相同标识的证书了。如果有安全上的需要，作废的标识还可以通过不同方式对外公布。证书作废过程的具体细节可以参见2.2.4证书的作废一节。

3.3.2 生产部

生产部负责加工认证卡，将证书数据写入认证卡中。生产部在接收到完整的证书数据后，通过专门的设备将证书写入认证卡。并将生产的认证卡递交给发放部。

3.3.3 发放部

发放部负责认证卡的发放工作。具体的工作过程参见4.3证书的发放一节。

3.3.4 系统信息管理部

系统信息管理部负责资料库中椭圆曲线参数和公钥因子矩阵的相关信息（比如状态）的添加，修改，检查ID证书所用的椭圆曲线参数和公钥因子矩阵是否有效，如果无效，则应该向密钥管理中心提出更换椭圆曲线或者公钥因子矩阵的请求。

3.4 资料库

CPK系统中，资料库（Repository）作为一个独立的模块，统一存储公钥因子矩阵、终端实体标识等数据。并向系统中其他模块提供访问这些数据的接口。资料库分别向内部和外部提供访问接口。对内的访问接口主要面向RMC和管理实体，而对外的接口主要面向终端实体；对内的接口是可读写的，而对外的接口是只读的。

资料库中存储的数据在逻辑上构成三种不同的实体：终端实体、安全网络和用户。一个终端实体拥有一个标识且只拥有唯一的一个标识，因此终端实体和标识是一一对应的，可以将标识看作终端实体的关键属性。安全网络由多个终端实体构成，这些终端实体共享它们所属安全网络的椭圆曲线、密钥因子矩阵和公钥映射算法，因此互相之间可以进行加密、认证的安全通信。资料库中的终端实体可以看作是CPK系统中物理存在的终端实体的逻辑形式。一个CPK系统可以只有一个安全网络，但是在分布式的CPK系统中，通常存在多个安全网络。用户是申请认证卡的人，一般情况下每个用户都有一个认证卡，即有唯一的标识，对应唯一的终端实体。但是用户也可能拥有多个认证卡，或者由于证书作废等原因暂时没有标识，导致用户对应的终端实体数量不是1，这是将终端实体和用户分离开来的原因。

3.4.1 资料库的部署

资料库存储了系统中大量敏感信息，存储敏感信息的资料库部分应该部署在防火墙保护下的系统内部网中，和Interent隔离。资料库应该加强访问控制，对内的接口只提供给RMC，但是RMC没有安全网络的写操作权限，RMC必须向资料库提供相关KMC签名的操作命令才可以对安全网络进行写操作。RMC本身具有对属于它的安全网络中终端实体和用户的全部读写权限，但是对其他安全网络中的终端实体的读写需要更高层中心的辅助。RMC和资料库需要VPN连接，当存在多个RMC访问同一个资料库时，资料库需要对RMC进行认证并对权限进行管理。如果需要提供作废证书列表等对外接口，则相关的数据可以部署在其他的服务器上，和资料库中其他数据相分离。对外的服务器可以联入Internet。

资料库可以通过关系数据库来存储数据，如Oracle等商业数据库，如果负载比较小，也可以采用MySQL等数据库。资料库根据需要还可以部署在LDAP数据库上，通常提供给内部访问的数据存储在关系数据库中，而提供给外部访问的数据可以存储在LDAP上。

3.4.2 资料库的数据设计

用户表

1. 用户编号
2. 用户姓名
3. 证件类型
4. 证件号码

5. 安全等级
6. ….

用户表的具体格式可以参照4.2节中注册申请表中包含的表项来确定。

终端实体表

1. 终端实体编号：CPK系统中的每个终端实体都有一个唯一的编号
2. ID：终端实体的标识，用于映射公钥的，是一个字符串
3. 公钥：根据当前矩阵和映射算法映射得到的公钥，固定长度字符数组，应该考虑是否包含此项。
4. 私钥：一般情况应该不包含此项。
5. 状态：标志当前实体状态，枚举类型，可能值：有效、作废、暂停、正在注册等
6. 开始时间：是一个日期时间对
7. 作废时间：同上
8. 用户编号：用于关联用户表中和终端实体对应的用户的相关信息。

安全网络表

1. 安全网络编号：CPK中的每个安全网络的编号都不同，也可以用ID来表示
2. 安全网络名称：
3. 安全网络所属RMC名称：
4. 状态：有效或作废
5. 开始时间：
6. 作废时间：

公钥因子矩阵表

1. 公钥因子矩阵编号：可以将矩阵的值得散列作为编号，也可以用唯一的整数
2. 散列值：防止篡改。
3. 矩阵：公钥因子矩阵是一个复杂的数据格式，包括密钥长度、行长度、列长度、矩阵值。
4. 椭圆曲线参数：
5. 映射算法：
6. 状态:：有效、作废、等待投入使用…等等
7. 安全网络编号：用于关联安全网络，可以根据该项查找一个安全网络的当前有效共钥因子矩阵。
8. 开始时间：
9. 作废时间

也可以将整个公开参数表保存在数据库中，而不是分开的矩阵。

3.4.3 资料库的对内接口

资料库的对内接口全部面向RMC，只能由RMC访问。

编号	1 建立安全网络
说明	
输入	公开参数表（定义参见2.3.2.7.2，是由KMC生成的）
输出	“成功”和新建安全网络的名称、编号　　如果失败返回“失败”

执行	1. 资料库检查RMC传来的公钥参数表的格式和KMC对该表的签名 2. 资料库在公钥因子矩阵表中新插入一个表项 3. 资料库在安全网络表中新插入一个表项 4. 返回该新建安全网络的编号和名称和“成功”

编号	2 修改安全网络
说明	主要是对公钥因子矩阵和椭圆曲线参数的替换
输入	公开参数表（定义参见2.3.2.7.2，是由KMC生成的），安全网络编号或名称
输出	“成功”或“失败”
执行	1. 检查RMC传来的公钥参数表的格式和KMC对该表的签名 2. 在公钥因子矩阵表中新插入一个表项并和相应安全网络关联，数据由公开参数表生成 3. 将原来安全网络对应的公钥因子矩阵表项的状态改为“作废”，并加入作废时间 4. 返回“成功”

功能	3 作废安全网络
说明	停止一个安全网络的使用
输入	安全网络编号
输出	“成功”或“失败”
执行	1. 将对应的安全网络状态改为“作废” 2. 将对应的矩阵表项状态改为“最终项” 3. 返回操作结果

功能	4 新建终端实体
说明	通过本接口可以在相应的安全网络中加入一个新的终端实体，包括一个新的标识。KMC在申请证书之前需要验证证书中的标识是否有效，如果所在的安全网络中已经存在一个同名的标识，或者存在一个已经作废的同名标识，那么这个标识不可使用，是无效的。KMC可以通过本接口新建一个使用该标识的终端实体，如果输出结果为“失败”，那么就说明该标识是无效的。
输入	安全网络编号，标识，终端实体初始状态
输出	如果成功返回“成功”和终端实体编号，如果失败返回“失败”
执行	1. 检查指定安全网络中是否有同名标识冲突，如果冲突，返回“失败”。 2. 检查制定安全网络中是否有冲突的作废标识，如果冲突，返回“失败”。 3. 在指定安全网络对应的终端实体表中添加一个新的终端实体，标识为输入的标识，状态为输入的状态。返回“成功”和该终端实体表项的编号。

功能	修改终端实体（修改状态，作废）
说明	
输入	安全网络编号，标识（或终端实体编号），更新后的终端实体状态
输出	如果成功返回“成功”，如果失败返回“失败”
执行	根据输入检查是否存在相应的终端实体，如果不存在返回“失败” 将终端实体的状态修改为输入的状态，返回“成功”

功能	查询终端实体
说明	
输入	安全网络编号，标识（或终端实体编号）
输出	终端实体数据格式（定义参见）或“失败”
执行	检查是否存在相应的终端实体，如果不存在，返回“失败” 返回终端实体的全部数据

功能	作废终端实体（加入作废标识库）
说明	
输入	终端实体编号，标识，安全网络编号
输出	
执行	根据输入找到相应的终端实体表项 将终端实体状态改为“无效”

3.4.4 资料库的对外接口

资料库对外的接口主要是指面向终端实体的接口。

标识的作废信息是资料库向终端实体提供的最重要的服务。在一些对安全性要求比较高的系统中，一个终端实体在和另一个终端实体建立安全通信之前，必须验证对方的标识是否有效，这就需要到公开客访问的资料库检查对方标识的状态，如果对方的标识已经作废，那么就不能使用该标识对应的公钥了。

资料库还可以提供一些标识查询的服务，但是这些服务不是CPK系统必须实现的，生产商可以根据应用自由提供相应的接口。

面向终端实体的接口是可选的，在一些应用中不需要提供标识验证服务。例如，安全性要求比较低的应用中不需要验证标识是否有效；移动通信应用中，通常无法在通信验证标识的有效性；在信用卡等应用中，标识的作废随着认证卡的作废而不再有效。这些应用中资料库都不需要提供面向终端的接口。

面向终端实体的接口可以有多种形式：

- CRL
- LDAP
- OCSP

3.4.5 资料库的相关协议

访问数据库的协议：

TCP连接，对于RMC来说，就好像使用本地的函数一样，因此在本地是不是应该有一个客户端呢？

客户端模块负责提供通信的细节，向KMC提供函数一样的功能。

数据格式可以仿照TCP/IP的数据包的格式

请求：

- 接口号
- 接口参数
- KMC的标识

- KMC的签名

应答：
返回的成功代码或失败代码

RMC和资料库通过TCP进行连接，通信是请求-应答方式的。RMC向资料库发送操作请求，资料库进行相应操作，并返回操作结果。RMC和资料库按照2.3.4.4定义的接口进行通信，本节介绍具体的数据包格式和通信协议。
请求数据包格式

```
RepositoryRequest ::= SEQUENCE{
    operCode            OperationCode,
    operParam           OperationParameter
}
```

3.5 ID证书

ID证书是CPK系统中的关键数据结构，CPK安全网络中的终端实体借助于ID证书实现加密、认证、签名等活动。在CPK系统中，ID证书是由密钥管理中心生成、并由注册管理中心封装到物理存储体中。ID证书中最重要的元素是用户的标识和用户的私钥，用户标识，即用户ID，是网络中实体的身份的全局唯一的逻辑表示，在CPK系统中每个标识都可以映射到唯一的一个公钥，因此ID证书同时可向用户提供公私钥对。

3.5.1 ID证书格式

ID证书的ASN.1抽象定义如下：

```
IDCertificate ::= SEQUENCE{
    version             Version DEFAULT v1, -- 证书版本号
    serialNum           IMPLICIT UniqueIdentifer,       -- ID证书的唯一编号
    keyUsage            KeyUsage,           -- 规定证书中私钥的用途
    validity            Validity,           -- 证书的有效期
    keyLength           KeyLength,          -- 证书中私钥的长度
    mapAlgorithm        MapAlgorithms,      -- CPK映射算法定义

    userID              IMPLICIT UniqueIdentifer,   -- 用户ID
    ellipticCurve       EllipticCurve,              -- 证书所采用的椭圆曲线参数
    protectedPrivateKey ProtectedPrivateKey,        -- 加密保护的用户私钥
    publicMatrix        PublicMatrix,               -- 公钥因子矩阵

    extension           Extensions OPTIONAL,        -- 证书扩展

    issuer              Issuer,                     -- ID证书的签发者
    signatureAlgorithm  SignatureAlgorithm,         -- 签名算法
    signatureValue      OCTET STRING,               -- 证书签名
}
```

```
Version ::= ENUMERATED {  v1(1) }
```
当前证书格式的版本为v1，不同版本的证书格式会有所不同。。

```
UniqueIdentifer ::= OCTET STRING
```
用于表示唯一的标识符，可以是可读的字符串，也可以是二制进字节序列。
在CPK的ID证书中，userID一般是可读的字符串。

```
KeyUsage ::= BIT STRING {
    decryptionOnly      (0),
    signatureOnly       (1),
    keyAgreement        (2)
}
```
KeyUsage所规定的私钥用途，包括签名、解密、密钥交换，私钥可以同时用于多个用途。KeyUsage由多个位构成，每个位表示某种功能是否可用。可以用固定长度的证书存储BIT STRING来减少存储空间。

```
Validity ::= SEQUENCE {
    notBefore       GeneralizedTime,
    notAfter        GeneralizedTime
}
```
validity规定了证书的有效期，其中包含notBefore和notAfter两个时间。GeneralizedTime为ASN.1中的标准时间类型，用4个字符表示年，月、日、时、分、秒用2个字符表示，其中小时为24进制。GeneralizedTime表示的是格林威治时间，应注意不同于本地时间。程序在使用证书之前必须检查当前的格林威治时间是否在notBefore和notAfter之间，如果否，则证书无效。

```
KeyLength ::= ENUMERATED {
    BIT-112(14),
    BIT-128(16),
    BIT-192(24),
    BIT-224(28),
    BIT-256(32),
    BIT-384(48),
    BIT-521(64)
}
```
KeyLength指定了ID证书中所包含的（椭圆曲线的）私钥的长度（以字节为单位）。

```
EncryptionAlgorithm ::= ENUMERATED {
    DES(0),
    3DES(1),
    AES128(2),
    AES192(3),
    AES256(4),
```

```
    UNDEF(200)
}
```

```
SignitureAlgorithm ::= DigestAlgorithm
```
签名算法总是采用ECDSA，其中的摘要算法采用DigestAlgorithm中规定的算法。

```
DigestAlgorithm ::= ENUMERATED{
    MD5(0),
    RIPEMD160(1),
    SHA1(2),
    SHA256(3),
    SHA384(4),
    SHA512(5),
    UNDEF(200)
}
```

```
ShuffleAlgorithm ::= ENUMERATED{
    PMT(0),
    UNDEF(200)
}
```

EncryptionAlgorithm、DigestAlgorithm、SignatureAlgorithm和ShuffleAlgorithm用于表示密码学算法。证书格式中包括预定义的一组算法，如AES128、SHA1等，如果应用系统需要采用其他的密码学算法，要保证算法的标识在UNDEF之后，即大于200。

```
MapAlgorithms ::= SEQUENCE {
    digestAlgorithm         DigestAlgorithm,
    encryptionAlgorithm     EncryptionAlgorithm,
    shuffleAlgorithm        ShuffleAlgorithm,
    mapParameter            OCTET STRING
}
```
映射算法用于将用户ID映射为密钥时所使用的各种（散列、加密、混洗）算，具体的定义参考其他章节。映射算法中需要用散列、加密和混洗算法，并且这些算法及其密钥是可变化的。MapAlgorithms中定义了散列算法digestAlgorithm、加密算法encryptionAlgorithm、混洗算法ShuffleAlgorithm和加密密钥mapParameter。

```
EllipticCurve ::= SEQUENCE {
    eCurveID        IMPLICIT UniqueIdentifier,
    eCurveValue     ECParamters OPTIONAL
}
```
在ID证书中可以指定证书所采用的椭圆曲线密码算法的参数。CPK系统为的系统中使用的每组椭圆曲线参数分配一个唯一的序列号eCurveID，椭圆曲线参数的具体值eCurveValue在证书中是可选的，可以仅仅通过eCurveID在证书之外找到具体的值。

椭圆曲线参数的具体值ECParameters的定义如下：

```
ECParameters ::= SEQUENCE {
    p           INTEGER,
    a           INTEGER,
    b           INTEGER,
    G           ECPoint,
    n           INTEGER
}

ECPoint ::= SEQUENCE {
    x           INTEGER,
    y           INTEGER
}
```

注：在ECParameters与ECPoint使用的整数INTEGER为大整数，与所采用的椭圆曲线密码算法的密钥长度相同，其具体的编码方式见本章3.1.2.4节。

```
ProtectedPrivateKey ::= SEQUENCE{
    encAlgorithm        EncryptionAlgorithm,
    encPrivateKey       OCTET STRING,
    encRandomNum        OCTET STRING,
    validateParameter   OCTET STRING
}
```

私钥在证书中是以随机数R1为对称加密密钥加密保存的，加密算法由encAlgorithm指定，encPrivateKey是加密后的私钥。

encRandomNum和validateParameter用于保护和验证私钥。

计算方法如下

encPrivateKey = $E_{R1}(pK)$，pK为用户私钥，R1为随机数（对称密钥），用于保护ID证书中的私钥。

encRandomNum = $E_{PWD}(R1)$，PWD为保护ID证书的用户口令。

validateParameter = $E_{R1}(R1) \oplus R1$，用于合法性验证。

```
PublicMatrix ::= SEQUENCE{
    MatrixID        IMPLICIT UniqueIdentifier,   -- 公钥因子矩阵的唯一编号
    ForkPoint       ECPoint OPTIONAL,
    matrix          Matrix
}
```

由于在CPK系统中，公钥因子矩阵需要占用一定的空间，有时受具体应用环境的限制，可能无法为每一级网络中的每一类ID（见本章第5节“CPK体系的扩展”）分别生成一个专用的公钥因子矩阵，此时不同类型的ID需要共用同一公钥因子矩阵。

具体的做法是，对某类ID，定义椭圆曲线上的一个点（ForkPoint），将其加上（点加）公钥因子矩阵中的每个因子得到一个新的公钥因子矩阵，新公钥因子矩阵就用于产生此类ID所对应的公钥。而用户对应的私钥也应加上与该点相对应的大整数。这样就可以将不同ID生成的公、私钥对分隔开。

在这种情况下，ID证书在ForkPoint中保存需要加的这个椭圆曲线上的点。而matrix中存储真正的公钥因子矩阵。

```
Matrix ::= CHOICE{
    applicationAdmin     BOOLEAN DEFAULT FALSE,
    matrixPath           MatrixPath,
    pointMatrix          ECPointMatrix
}
```

在CPK中，对公钥因子矩阵可以有多种存放机制：

- 应用程序负责公钥因子矩阵的存放和访问，在ID证书中不定义，在这种情况下applicationAdmin的取值为TRUE；
- 本证书的ID与其他类型的ID共用同一的公钥因子矩阵（通过ForkPoint），而公钥因子矩阵存放在其他证书中，此时就需要在MatrixPath指明存放共用的公钥因子矩阵的ID证书的唯一标识标识及矩阵在ID证书中的偏移量，具体见MatrixPath的定义。
- 公钥因子矩阵就保存在本ID证书中，其格式见ECPointMatrix的定义。

```
MatrixPath ::= SEQUENCE{
    serialNum            CertSerialNum,
    offsetInCert         INTEGER,
}

ECPointMatrix ::= SEQUENCE {
    matrixRowNum         INTEGER,                  -- CPK公钥因子矩阵的行数
    matrixColumnNum      INTEGER,                  -- CPK公钥因子矩阵的列数
    points               SEQUENCE OF ECPoint       -- 矩阵中的公钥因子
}

Issuer  ::= SEQUENCE {
    issuerID         IMPLICIT UniqueIdentifier,
    issuerMatrix     PublicMatrix
}
```

Issuer明确定义了证书签发者的唯一标识issuerID，并指明了签名所用ECC密钥所属的公钥因子矩阵。

```
Extensions ::= SEQUENCE SIZE(0..MAX) OF Extension

Extension ::= SEQUENCE {
    extnID           OBJECT IDENTIFIER,
    critical         BOOLEAN DEFAULT FALSE,
    extnValue        OCTET STRING
}
```

为了提供良好的可扩展性，ID证书定义了扩展段Extensions作为扩展ID证书用途的接口，各应用可根据实际需要自行进行扩展。

如在电子银行等应用中，常需要对用户划分不同的密级，则可在Extension中定义具体含义

与应用相关用户等级密钥UserGradeKey，将用户划分为多级，如：顾客级、职工级、中层领导级和高层领导级，等级密钥均为对称密钥，向低级兼容，如：第*n*级用户的ID证书中包含1～*n*级的所有密钥。

3.5.2 ID证书实例

编码标准
ASN.1定义的证书格式可以通过多种编码标准来实现。这里定义了IDC编码标准，用于手机等存储空间有限的设备上ID证书的编码。

基本数据类型的C语言编码标准
本编码标准遵守Intel386体系结构下字节编码顺序和Win32平台下C语言的数据结构。ASN.1表示的数据类型最终映射为C语言的数据类型及其组合。包括unsigned char、unsigned short、 unsigned long和字节数组。

ENUMERATED的编码
ENUMERATED编码为1byte，因此枚举的类型不能超过256个

BIT STRING
BIT STRING编码为1byte，其中比特顺序从低端排起。

INTEGER的编码
INTEGER在不同大小时可选编码为unsigned char, unsigned short, unsigned long。
还可以编码为由系统指定长度的字节序列，字节序列中存储的是16进制表示的整数，其中字节序列的最左侧字节是整数的最高位字节。这种编码不保存整数所占字节序列的长度，因此应用证书的系统必须知道INTEGER的长度。

OCTET STRING的编码
OCTET STRING由表示长度的INTEGER和字节序列构成。

SEQUNCE的编码
SEQUNCE数据类型包含多个子元素，编码时直接对其子元素进行编码。

CHOICE的编码
CHOICE类型的属性编码时头部有1byte的标志用于表示其子元素的类型。标志的值表示其子类型在定义中的顺序，顺序的编码从零开始。

OPTIONAL的编码
如果一个属性有OPTIONAL标志，那么在对该属性编码前，首先增加1byte的标志用于表示该属性是否存在，如果标志的值为零则该属性不存在，后面的数据是属于下一个属性。否则按照正常的方式在后面添加该可选属性的编码数据。

CertSerialNum的INTEGER编码为字节序列，长度为6
ECParameters和ECPoint的INTEGER编码为字节序列，长度由keyLength规定

rowNum和columeNum这两个INTEGER编码为unsigned char
offsetAddress编码为unsigned long

userID编码为unsigned char和字节序列

Attribute	DataType	Value
version	byte	1
certUsage	byte	0x02 (signatureOnly==TRUE)
Validity		
notBefore	byte[14]	“20040918121036”
notAfter	byte[14]	“20050918121036”
keyLength	byte	24 (BIT-192)
mapAlgoithm		
digestAlgorithm	byte	2 (AES128)
encryptionAlgorithm	byte	3 (SHA256)
mapParameter	byte[16]	(AES BlockSize == 16bytes)
userID	byte	19,
	byte[19]	“Johnson@hotmail.com”
eclliptic Curve		
eCurveID	byte[6]	0x00 00 00 00 00 05
ProtectedPrivateKey OPTIONAL	byte	0x00 (NULL)
encAlgorithm	byte	3(AES128)
encPrivateKey	byte[32]	(32bytes)
encRandomNum	byte[16]	0x----
validateParameter	byte[16]	0x----
publicMatrix		
baseMatrixID	byte[6]	0x00 00 00 00 00 02
OPTIONAL	byte	0x00 (offset == NULL)
Offset		
OPTIONAL	byte	0xff (matix != NULL)
matrix		
rowNum	byte	32
columeNum	byte	32
value		
CHOICE	byte	2 (points)
points	byte[len]	(len = rowNum*columeNum *keyLength*2
ECPoint.x	byte[24]	(24 == keyLength)
ECPoint.y	byte[24]	
----------	----------	other ECPoints
extension OPTIONAL	byte	0

issuer		
issuerID	byte	3
	byte[4]	“KMC”
issuerMatrix		
baseMatrixID	byte[6]	0x00 00 00 00 00 02
offset OPTIONAL	byte	0
matrix OPTIONAL	byte	0
signatureAlgorithm	byte	ECDSA
signatureValue	byte[24]	

3.6 协议与标准

CPK系统中的通信可以有联线的和数据报的，如手机通信和Email通信。

不同类型的通信采用不同的通信协议

同时在线的可以采用密钥交换协议交换会话密钥，然后通过对称密钥通信。

单方向发送数据报，必须通过公钥加密算法和对称加密算法

3.6.1 对称加密标准

对称加密用于加密数据报、会话通信。通信双方共享一个对称密钥和相同的对称加密算法。和固定使用椭圆曲线公钥算法不同，CPK中可以根据不同的应用选择不同的对称加密算法，如DES，3DES，AES等。这些算法可以参见其相关标准。

3.6.2 口令密钥标准

在对称加密中，可以利用用户输入的口令生成对称加密密钥。Kerberos中提出了将口令转化为64bit DES密钥的方法，并且用CBC模式对数据进行加密。

在CPK中，对私钥、随机生成的对称加密密钥等数据量很小的数据进行口令加密。而当数据量比较大，可以随机生成对称密钥，用强度比较高的算法如AES对其进行加密，在对随机密钥用口令加密，附加在加密数据的后面。

3.6.3 公钥加密标准

CPK中采用ECAES标准作为其公钥加密标准。

首先，假设接收实体*B*拥有域参数 $D=(q,FR,a,b,G,n,h)$ 和公钥 $Q_B=d_AG$ 。同时，假设发送者A拥有的 D 和 Q_B 可信拷贝。在下列描述中，MAC表示HMAC等消息认证码（MAC）算法，ENC表示三重DES、AES等对称加密算法，KDF表示从共享秘密点推导出密钥的密钥推导函数。

要加密发给实体B的消息 m ，A需要执行：

1．在范围$[1,n-1]$随机地选择一个整数r。

2．计算$R=rG$。

3．计算$K=hrQ_B=(K_x,K_y)$，检查$K\neq O$。

4．计算$k_1\|k_2=KDF(K_x)$。

5．计算$c=Enc_{k_1}(c)$。

6．计算$t=MAC_{k_2}(c)$。

7．将(R,c,t)发给B。

要解密密文(R,c,t)，B执行：

1．对R执行部分密钥验证。

2．计算$K=hd_BR=(K_x,K_y)$，检查$K\neq O$。

3．计算$k_1\|k_2=KDF(K_x)$。

8．验证$t=MAC_{k_2}(c)$。

9．计算$m=MAC_{k_2}^{-1}(c)$。

在加密和解密过程中时间开销较大的操作是椭圆曲线的标量乘法运算。

3.6.4 数字签名标准

ECDSA模式是椭圆曲线系统中的签名、验证算法。首先，假定签名者A拥有域参数$D=(q,FR,a,b,G,n,h)$和公钥$Q_A=d_AG$。同时，假定B也拥有D和Q_A的可信拷贝。在下面描述中SHA-1表示160比特散列函数。

要对消息m进行签名，A执行下列操作：

1．在范围$[1,n-1]$随机地选择一个整数k。

1．计算$kG=(x_1,y_1)$，以及$r=x_1 \mod n$。如果$r=0$，转第1步。

2．计算$k^{-1} \mod n$。

3．计算$e=SHA-1(m)$。

4．计算$s=k^{-1}\{e+d_A\cdot r\} \mod n$。如果$s=0$，转第1步。

5．A对消息m的签名即为(r,s)。

要验证A对消息 m 的签名 (r,s)，B执行下列操作：

1．验证 r 和 s 为范围 $[1,n-1]$ 的整数。

2．计算 $e=SHA-1(m)$。

3．计算 $\omega=s^{-1} \mod n$。

4．计算 $u_1=e\omega \mod n$ 和 $u_2=r\omega \mod n$。

5．计算 $u_1G+u_2Q_A=(x_1,y_1)$。

6．计算 $v=x_1 \mod n$。

如果 $v=r$ 就接收此签名。

3.6.5 PMT算法

置换表

置换表由16行16列构成，列是置换轮，行是置换的起译点。如果指示吗为（07，05）则指07列（7置换伦）第5起点。

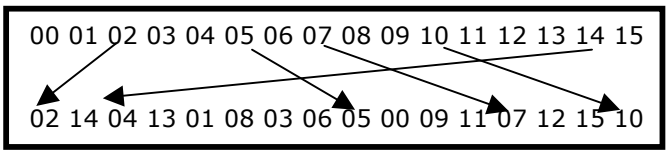
一个密钥变量8字节，每一字节的高4位指示所选置换轮，低4位指示置换起点，一共进行8次置换变换，最终得到新的置换序。

置换序的选择，每一次运算在16个列中取8种，而每一列都由16个起点，因此其变化量是足够大的。

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	03	07	04	10	08	02	05	11	14	09	12	06	09	05	13	08
01	06	15	05	05	13	06	08	07	03	13	07	03	12	15	04	03
02	14	05	11	08	07	10	13	12	05	10	14	10	15	04	14	10
03	11	10	09	12	05	13	09	15	06	15	13	05	04	01	06	15
04	13	01	07	14	11	12	01	10	13	07	02	00	07	14	15	01
05	07	13	14	09	00	03	12	02	00	02	09	02	01	08	08	06
06	15	00	08	00	02	11	02	14	12	01	00	11	08	02	01	00
07	10	12	13	06	15	01	15	04	02	08	03	14	13	11	03	11
08	02	03	12	01	01	04	10	13	11	12	11	01	03	12	12	04
09	00	06	02	13	06	14	00	01	01	03	06	13	05	07	05	02
10	04	14	15	15	04	15	14	08	04	14	04	07	14	09	11	12
11	01	08	06	03	14	05	04	03	09	05	01	15	06	13	09	05
12	09	04	00	04	03	09	07	06	15	00	15	08	02	03	00	14
13	12	09	10	11	09	08	03	05	07	04	10	04	10	00	07	09
14	05	02	03	07	12	00	06	00	08	06	05	09	00	10	10	07

15 08 11 01 02 10 07 11 09 10 11 08 12 11 06 02 13

选择不同置换序、不同起点，都能组成新的不同的置换关系。起点和轮序由密钥确定，在一轮变换中，假设轮序为第7论，起点为第5起点，那么形成下面置换关系：



程序实现

```
program permute32(input,output);
type onedim=array [0..15] of integer;
      twodim=array [0..15,0..15] of integer;
var i,j,k:integer;
    a,b,c,d:integer;
    disk:twodim;
    key,dka,dkb,dkc,dkd,ee:onedim;
    fil:text;
procedure diskchange(i,j:integer; dka:onedim; var dkb:onedim);
        var p,k:integer;
        begin
            for k:=0 to 15 do
            begin
                    p:=(i+k) and 15;
                    p:=disk[p,j];
                    p:=p-i;
                    if p<0 then p:=p+16;
                    dkb[p]:=dka[k];
            end;
        end;
        begin
            for i:=0 to 15 do begin dka[i]:=i; dkb[i]:=i+16; end;

            for i:=0 to 15 do
            begin a:=key[i] div 16;
                    b:=key[i] mod 16;
                    diskchange(a,b,dka,dkc);
                    for j:=0 to 15 do dka[j]:=dkc[j];
                    c:=key[i+1] div 16;
                    d:=key[i+1] mod 16;
                    diskchange(c,d,dkb,dkd);
                    for j:=0 to 15 do dkb[j]:=dkd[j];
                    i:=i+1;
                            for j:=0 to 7 do    ee[j]:=dka[j];
```

```
                for j:=0 to 7 do    dka[j]:=dka[j+8];
                for j:=0 to 7 do    dka[j+8]:=dkb[j];
                for j:=0 to 7 do    dkb[j]:=dkb[j+8];
                for j:=0 to 7 do    dkb[j+8]:=ee[j];
            end;
        end.
```

置换结果

给定数据data＝

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

给定密钥key＝

7B C9 6F 1B 21 6C BB E7 DE C7 62 4F 2D 71 EA 60

置换结果：

1圈：04 0d 01 0e 0a 06 0c 09 1c 15 1e 17 10 1a 19 1d

11 1f 12 18 1b 14 16 13 03 05 07 0b 0f 08 02 00

2圈：0c 1a 04 19 0e 17 01 06 13 1b 11 0f 12 00 05 0b

08 1f 14 02 03 16 07 18 1d 1c 1e 09 10 0d 0a 15

3圈：1a 00 17 19 13 0b 0e 04 16 0a 08 1e 18 1c 09 1f

03 10 1d 15 07 14 0d 02 12 01 11 0c 06 05 1b 0f

… …

8圈：0c 0d 07 01 1f 10 09 1d 03 1c 15 0a 19 11 0b 1a

00 02 16 18 06 08 0f 12 05 0e 13 1e 04 1b 17 14

自然序通过8圈置换变换，得到新的置换序：

0c 0d 07 01 1f 10 09 1d 03 1c 15 0a 19 11 0b 1a

00 02 16 18 06 08 0f 12 05 0e 13 1e 04 1b 17 14

3.6.6 在线认证协议

计划仿照SSL设计一个相应的认证协议

3.6.7 密钥交换协议

密钥交换协议DH2（ECDH的改进协议）的基本思想是根据实体*A*在另一实体*B*的公钥的基础上产生一个共享密钥，所以如果双方实体如果同时执行以对应密钥为输入的密钥交换协议，那么他们将能够恢复出相同的共享秘密。假设实体*A*的域参数为$D=(q,FR,a,b,G,n,h)$，并且私钥为d_A。同时假定实体*B*拥有与域参数D相对应的公钥$Q_B=d_BG$。在此，要求公钥Q_B至少是部分有效的。

则密钥交换协议DH2的步骤如下：

1．*A*选择随机数*r*，计算 $P = r(d_B G)$，发送给*B*。

2．*A*计算 $rG = (x_P, y_P)$，将 $k = x_P$ 作为密钥。

3．*B*用自身私钥计算 $d_B^{-1} P = d_B^{-1}(r(d_B G)) = rG$，同样将 $k = x_P$ 作为密钥。

由于只有实体*B*拥有 d_B，只有他能从 $P = rd_B G$ 中恢复出 $rG = (x_P, y_P)$，所以实体*A*与*B*之间能够拥有共享秘密，而其他人则无法获知该秘密。

3.7 系统的管理

系统中的功能模块出于安全目的，需要进行访问控制、日志记录和备份等管理。本节统一说明这些管理功能的规格。

3.7.1 访问控制

每个部署了系统功能模块的计算机都需要一个管理系统来进行访问控制。管理员需要向管理系统输入用户名和口令才能启动功能模块。通常功能模块本地存储的私密数据需要通过功能模块的公钥进行加密，加密标准参见[　　]。功能模块的私钥数据通过管理员的口令，以口令加密标准加密存储。因此只有管理员才能操作相应的功能模块。管理员输入口令后，口令或者功能模块私钥是以明文的形式保存在内存中的，功能模块进行各种操作不再需要管理员输入口令。当系统退出，或者管理员明确指定时，系统必须将内存中的私密信息消除，同时要注意清除残留在操作系统内存交换文件中的数据。

3.7.2 日志系统

功能模块进行的全部操作都需要记录进日志中，用于检查和错误恢复。

日志具体格式参见下表：

时间戳
操作者
操作命令
输入参数
输出（如果输出文件，记录存储路径）

3.7.3 备份系统

系统中的日志、私密信息、生成的文件等需要定期进行备份。可以将这些数据备份到磁带或光盘上，采用公钥加密模式进行加密。

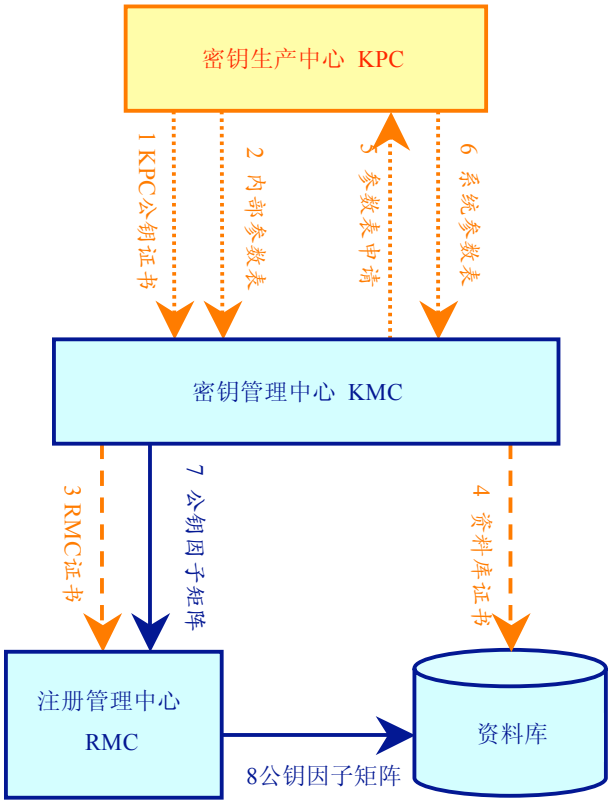
3.7.4 安全通信

功能模块之间通信之前应该进行认证，通信数据要加密、防篡改。功能模块通过证书进行安全的通信，因此提供证书服务的CPK系统也是一个应用CPK证书的系统。

4. 密钥及证书管理

这里要对生命周期进行说明，并且总体上介绍一下本节是说明什么的

4.1 系统初始化



1. KPC首先生成KPC公私钥对，并根据公私钥对生成KPC公钥证书。KPC将公钥证书发放到KMC，如图中所示的数据流1“KPC公钥证书”，其中的短点划线表示必须通过物理信道发放。由于KPC是和网络隔离的，因此需要管理员将要发放的数据拷贝到目的地。也可以将该公钥证书硬编码到KMC中。图中的数据流2、5、6也采用这种安全的通信形式。
2. KPC将CPK系统的内部系统参数表发放到KMC。如图中数据流2“内部参数表”所示，内部系统参数表和2.3.4介绍的系统参数表格式完全一致，但是不是用于向终端实体发放证书，而是用于向系统内的功能模块发放证书。KMC将通过KPC的公钥证书验证该系

统参数表的真实性。

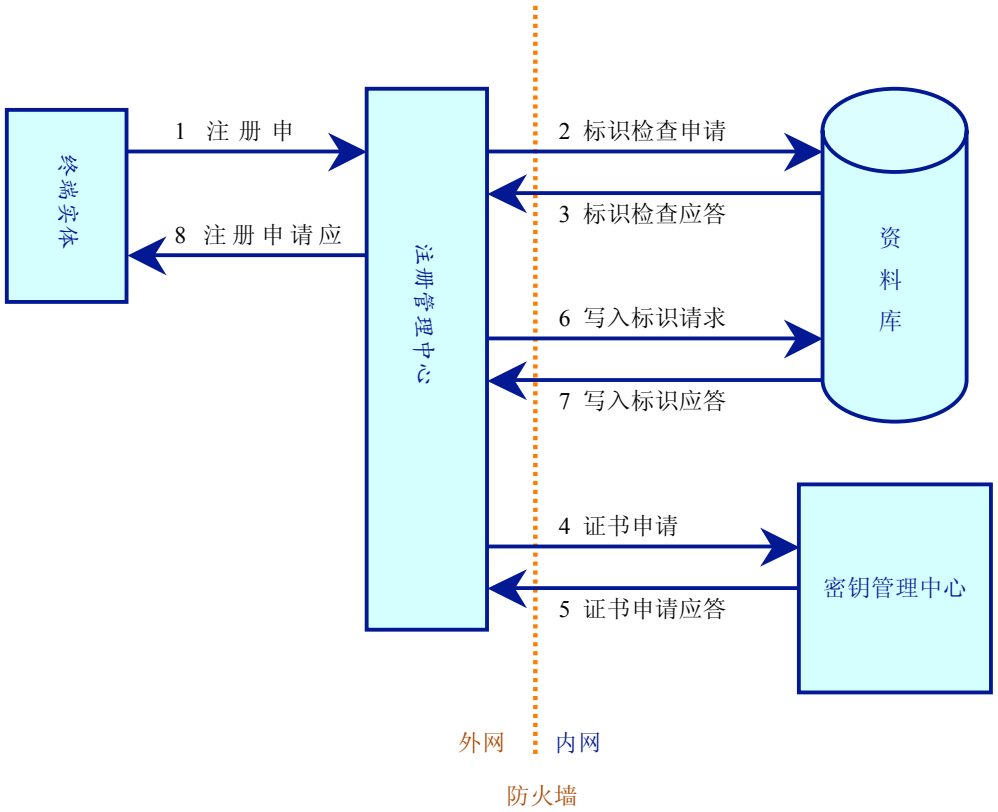
3. KMC通过接收到的内部参数表生成各个功能模块的证书，并发放给这些模块，如图中数据流3所示KMC向RMC房发放参数表，这里的长点划线表示要通过安全信道传送，可以通过网络，也可以不通过网络，由于此时RMC和资料库没有KMC的公钥，不能对这些数据进行验证，必须由系统中的管理员来控制证书的分发并对其验证。
4. KMC向资料库发放证书。注意3、4是同时发生的，而且KMC还会向自己发放一个证书。
5. KMC向KPC申请用于向外部发放证书的系统参数表。
6. KPC生成系统参数表,发放给KMC。
7. KMC获得参数表后就可以生成对外发放的证书了，但是KMC需要先通知系统中其他模块系统采用的参数，数据流7中的“公钥因子矩阵”中包括椭圆曲线参数和公钥因子矩阵，注意KPC发放的系统参数表中的私钥因子矩阵只能在KMC加密保存，不能对其他模块公开。
8. KMC不直接和资料库交互，由RMC将系统参数发放给资料库，并在资料库中建立一个安全网络。KMC还需要确定公钥影射算法。

初始化工作完成之后，RMC就可以接收终端实体的申请，提供证书的生成、发放和作废的服务了。

4.2 ID证书的申请

终端实体在加入CPK系统之前必须首先进行注册。如图中1所示，终端实体向本地的注册管理中心递交注册申请，注册管理中心、资料库和密钥管理中心将完成一系列步骤，最终生成一张具有证书的认证卡，由注册管理中心发放给终端实体，即图中8所示的“注册申请应答”。

证书申请图描绘证书申请过程，其中的箭头代表了申请过程中功能模块之间的数据流动。本节将综合数据的流动和功能模块内部处理过程两方面来说明证书申请中系统的工作过程。



4.2.1 终端实体向RMC提交注册申请

证书的申请过程是由终端实体向RMC发起注册申请开始的。
如果有法律方面的需要，注册管理中心可以首先向终端实体提供一份合同，双方签订合同以确定双方的责任和义务。

CPK系统中通常采用实名制原则，因此需要用户亲自到注册管理中心填写注册申请表。表2-xx给出了一个示例。具体的应用可以根据此表进行修改和扩充。

姓名	丁一	性别	男
出生日期	1980年1月1日	照片	
证件类型	身份证	证件号码	256701800101234
联系地址	北京大学	邮编	100089
电话号码	010-12345678	手机号码	13812343241
单位名称	软件学院	职务	招生办工作人员
单位电话	010-12345678		
标识类型	电子邮件	标识	ding1@pku.edu.cn
安全等级	绝密	填表日期	2004-8-1

4.2.2 RMC对注册申请进行审核

RMC在接收到终端实体的注册请求之后，需要对其进行审核。

审核的内容主要包括：

- 对身份的审核：　　需要用户提供身份证或其他有效证件。
- 对电话号码的审核：需要进行电话验证。
- 对住址邮编的审核：需要派出所证明材料。
- 对单位信息的审核：需要单位出具的证明材料。
- 对电子邮件的审核：需要在线的验证。
- 对安全等级的审核：用户申请的安全等级要和用户的身份适应。

如果申请没有通过审核，RMC应立即返回标记为“未通过审核”的应答，通知用户重新填写注册申请表。否则进入下一步。

4.2.3 RMC向资料库申请检查标识有效性

如果标识不是由密钥管理中心统一生成，那么可能出现申请的标识和系统中已有的标识冲突现象。因此RMC应该访问资料库，验证标识的有效性，如图中2所示。RMC使用了资料库的“新建终端实体”接口，在资料库中新建一个采用申请标识的状态为“无效”的终端实体。具体的接口定义可以参见3.4资料库一节。如果操作成功，说明标识没有冲突。否则需要返回错误信息，通知用户重新选择标识。

4.2.4 RMC向KMC申请证书

RMC确定了注册申请的有效性之后，通过KMC提供的接口向KMC申请生成证书。图中4所示的“证书申请”满足KMC接口的定义，具体格式参见3.2密钥管理中心一节。KMC在接收到RMC的请求后会检验申请的有效性，如果有效，就按照2.3.x中的定义生成证书并传回给RMC作为应答，否则传回标记为“失败”的应答。

4.2.5 RMC通知资料库将标识写入数据库

收到KMC生成的证书后，RMC可以确定证书和标识的有效性，因此可以通知资料库将该标识写入。实际上在4.2.3标识的验证过程中，随着终端实体的建立，标识已经被写入资料库中。因此只需通过资料库的接口“修改终端实体状态”将拥有该标识的终端实体状态从“无效”改为“有效”即可。先将标识写入资料库的做法是为了防止一个证书的申请过程中其他用户申请到相同的标识。

4.2.6 RMC生成认证卡

RMC必须负责认证卡的生成，如果需要委托其他的生产厂商将ID证书写入认证卡，必须由管理人员负责全程的监控，防止认证卡的生成过程中出现秘密泄漏。

4.2.7 RMC将认证卡发放给终端实体

RMC必须通过安全的物理信道将认证卡发放给终端实体，有两种可选的发送方式将：

对ID证书的发放过程有如下安全建议：

- ID证书的发放过程应该有详细的纪录
- 通过特快专递等传递方式应该接收用户的收到确认，如果在指定时间之内未收到认证，必须作废证书
- 用户到指定地点领取证书，应该有足够的身份证明
- 如果用户没有在制定时间范围内领取证书，必须收回证书

4.3 ID证书的产生

为了加强CPK系统的安全性和易用性，证书是通过认证卡这种物理形式实现并发布的。证书的生产是指将逻辑形式的证书文件转化为物理形式的认证卡。本节规定认证卡的物理属性和认证卡的生产规范

4.3.1 认证卡的物理属性

认证卡可以是智能卡、U棒、手机卡等设备模块，也可以直接写入物理设备的ROM中。充当认证卡的设别必须保证以下的必要条件：

- 认证卡必须有独立的处理器，能够利用私钥进行ECC的签名和解密。
- ID证书的用户私钥必须存储在能够防止外部读写的存储器上。一旦密钥管理中心将用户私钥写入认证卡上的这种存储器后，只有认证卡上的处理器才能够对其进行读写。
- 认证卡上的存储空间应可以容纳整个ID证书。
- 认证卡是不可复制的

认证卡的这些性质可以防止用户的私钥暴露。一旦注册管理中心将ID证书写入认证卡，私钥对于认证卡的外界就不再是不可读写的了，所有和私钥相关的处理必须在认证卡的内部进行，因此认证卡必须具有一定的处理能力。用户的私钥用于数字签名和解密经用户公钥加密后的数据，认证卡必须保证这两种处理能力。其他的和证书相关的运算都可以由认证卡外部的处理器完成。

认证卡在不同的应用中可以和其他的卡合而为一，例如在手机安全通信中，认证卡可以和SIM卡结合；在社保系统中，可以将身份信息印刷在认证卡体上，并将身份信息存储在认证卡的存储器中；在金融系统中，信用卡可以以认证卡的形式出现，可以在卡体上印刷信用卡号，提供用户的手写签名。

可以通过口令的方式加强对认证卡的保护。用户必须输入认证卡访问口令才可以访问认证卡。口令由系统或用户生成，必须保证一定的强度。

4.3.2 认证卡的生产规范

涉及到人员和设备的管理

输入：RMC生成的证书文件最终版
输出：认证卡

由注册管理中心的生产部负责证书的生产：

- 将证书的信息写入认证卡：详细
- 将部分信息印刷在认证卡表面

4.4 ID证书的发放

对证书以认证卡的形式发放，发放过程有如下安全建议

- ID证书的发放过程应该有详细的纪录
- 通过特快专递等传递方式应该接收用户的收到确认，如果在指定时间之内未收到认证，必须作废证书
- 用户到指定地点领取证书，应该有足够的身份证明
- 如果用户没有在制定时间范围内领取证书，必须收回证书
- 如果认证卡有口令，那么口令和认证卡必须以不同的信道传送给用户。

4.5 ID证书的使用

拥有ID证书的终端实体可以通过证书获得自己的公私钥对和安全网络中其它终端实体的公钥。终端实体可以利用证书中提供的公私钥信息进行三类主要的应用：

- 加密/解密：发送方用接收方的公钥加密报文，接收方能够用自己的私钥解密该报文且只有接收方能够解密该报文。
- 数字签名：发送方用他自己的私有密钥对报文进行数字签名。接收方能够根据签名验证报文确实是由发送方生成并且没有被第三方篡改。
- 密钥交换：两个终端实体分别利用自己的公私钥对交换会话密钥。

4.5.1 使用证书的终端实体

在CPK系统中，一个完整的终端实体由以下部分组成：

1) 应用模块：CPK系统通常是在一些原本没有安全功能的应用系统中加入安全功能，终端实体中的应用模块就是指原来的应用中能够互相通信的实体。例如手机、电子邮件软件等，这些实体在没有应用CPK系统之前不能进行安全通信。
2) 认证卡：存储证书数据，进行和私钥相关的运算
3) 读卡器：是认证卡和安全模块的接口，属于物理设备
4) 安全模块：认证卡虽然有一定的处理能力，但是功能比较弱，只负责和私钥相关的运算，其他的安全功能需要安全模块来完成。安全模块可以使一个软件模块，
5) 相关人员：

由于认证卡和安全模块软件是紧密结合的，因此需要进一步细化系统的功能、相关协议和操作才能很好的定义本节的内容。

4.5.2 加密/解密

如果终端实体Alice要向终端实体Bob发送加密数据，首先要求两者必须拥有同一个安全网络的认证卡。加密解密的过程如下：

Alice：

将认证卡插入读卡器。

输入认证卡口令，系统验证口令的正确性，并做出相应。

输入需要加密的数据。

系统随机生成一个用于对称加密的会话密钥k，

系统用密钥k和选定的对称加密算法E加密数据m生成加密数据Cm。

选择数据的放松对象，即Bob的标识。

系统根据Bob的标识映射到公钥因子矩阵，计算出Bob的公钥UK。

系统用UK加密会话密钥，附加到Cm后面。

将这些数据发送到应用模块中，由应用模块作为通信数据。

[todo]

4.5.3 数字签名和验证

[todo]

4.5.4 密钥交换

[todo]

4.6 ID证书的作废

4.6.1 证书作废的条件

由于安全性的原因，证书需要进行作废。

作废的原因主要分两种：标识作废和系统更新。由于单个证书的安全问题导致证书作废的，只要作废相应的证书即可，这种作废方式称为标识作废。而如果整个CPK系统出现安全问题，那么就需要对整个系统进行更新，相应的系统内的全部证书都需要作废。系统更新之后由于系统中的参数全部更新，原来的标识映射得到新的公私钥对，因此不用改变标识；而对于单个证书由于私钥暴露的作废，只能抛弃证书中的标识，因此称为标识作废。

（1）标识作废

由于安全或权限的原因，终端实体不能继续使用原来的标识，这时终端实体必须向RMC申请作废证书。下面列出了常见的标识作废原因：

- 私钥暴露：如果证书私钥暴露，那么对应的标识和公私钥对就都不能继续使用了。这时必须作废该标识，或者暂停使用这个标识，直到系统更新后才能重新使用该标识。认证卡可能在发放、使用的过程中丢失，这都有可能导致私钥暴露。
- 证书过期：在安全性比较高的应用中，为了防止终端实体的公私钥对被暴力破解，需要规定证书的有效期，过期的证书必须进行作废。
- 更换标识：终端实体作为标识的电话号码、邮件地址、计算机网络地址、职务名称等都有可能发生变化，在申请具有新标识的证书之前，必须作废旧的证书。

（2）系统更新

CPK系统可能进行整个系统的更新，包括系统参数（椭圆曲线、密钥因子矩阵、映射算法）和所有证书的更新。系统的更新一般是出于安全上的考虑，当整个系统面临安全性的问题总时，就必须进行系统更新了。系统更新是非常复杂的，要包括系统中全部证书的更新，

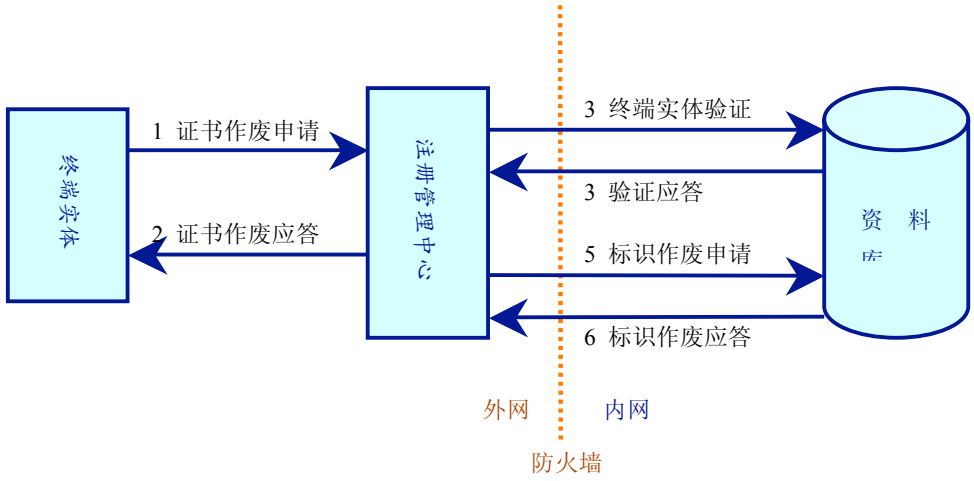
这在某些非常大型的应用中几乎是不能完成的任务。

可能导致系统更新的原因如下：

- CPK系统参数总有一个生命周期上限。在KPC为CPK系统生成的系统参数表中包含validity（有效性）一项，参见，有效性指明了使用该参数表的系统的生命周期长度，当系统的工作时间已经超过了有效期之后，那么系统就不再安全了，这是因为椭圆曲线密码系统有被破解的可能。
- CPK系统中的密钥矩阵的大小决定了系统中能够容纳终端实体的数量，如果在系统的运行过程中，发放的证书数量超过了系统容量，导致不同的标识映射到相同的公私钥对，那么必须对更新系统的参数。
- 在很多情况下CPK的生命周期要短于系统参数表上所标称的有效期，主要的原因是在系统的运行过程中，秘密信息有可能暴露。例如存储在KPC和KMC上的私钥因子矩阵一旦暴露，则系统中的私钥全部暴露。证书中的单个私钥暴露过多也危及系统的安全性，参见，攻击者可以利用暴露的多个私钥，通过求解方程的方式来破解出整个私钥矩阵。

4.6.2 作废的过程

系统更新需要重新执行初始化过程，可以参见。这里只描述标识作废的证书作废过程。如图所示：



首先由终端实体向注册管理中心递交“证书作废申请”。证书作废申请中一般包括证书序列号、标识和作废原因。根据不同的应用，终端实体可以通过多种方式递交证书作废申请：

- 通过网络在线作废证书：终端实体，终端实体和系统共享的秘密信息来作废证书。
- 通过电话作废证书：这种方式在很多应用中都比较方便快捷，例如在信用卡或电话SIM卡等形式的认证卡中，认证卡上印有比较长的认证卡编号，用户可以根据这个编号打电话到注册管理中心作废证书。
- 如果对安全性要求比较高，可以由证书所有者持有效证件到注册管理中心办理作废。有效证件可以是身份证或其他系统内认可的有效证件。

注册管理中心在接收到“证书作废申请”需要验证终端实体提供的身份证明信息与资料库中保存的信息是否一致，如果不一致，注册管理中心向终端实体返回标记为“无效”的证书作废应答。

如果验证一致，注册管理中心就向资料库发送“标识作废申请”，这个申请的格式可以

参考xxxxx中对于该接口的定义。

资料库接收到该申请后，首先将证书标识的状态修改为“作废”，然后可以根据应用的安全性要求，决定是否将该标识加入作废标识库。如果加入作废标识库，则终端实体可以通过各种方式访问到作废的标识，从而验证使用的标识是否有效。

资料库随后向注册管理中心返回标记为“成功”的标识作废应答，注册管理中心然后向终端实体返回证书作废应答。

4.6.3 作废标识的公布

系统对证书作废有三种处理方式：

- 如果认证卡有很高的安全性，即使丢失也不会暴露私钥数据，而且没有口令就完全不可用，那么丢失认证卡的用户只需要重新向注册管理中心申请一个和原认证卡具有相同标识和公私钥对的新认证卡。
- 如果认证卡不具有高的安全性，而且相应的应用不需要保持原有的标识，那么系统只需要作废证书中的标识，保证不会生产出相同标识的新证书即可。用户只需要到注册管理中心申请一个具有新标识的认真卡。例如在银行信用卡应用中，信用卡号作为标识是可以抛弃的，银行只要记录下作废的卡号就可以了。
- 在对安全性要求比较高的应用中，终端实体在通信之前需要验证所用标识的有效性，这不仅需要资料库标记作废的标识，还需要向终端实体公布作废的标识。

可以采用推或拉的方式通知用户作废的证书。在推的方式中，由CPK系统主动向用户提供作废证书信息，系统可以采用电子邮件或RPC等方式通知用户。在拉的方式中，如果用户有验证ID有效性需求的时候，向系统查询该ID是否作废。系统可以提供Web方式或服务的方式来提供作废列表。

已有比较成熟的技术可以用于作废证书的公告，如LDAP和OCSP。

5. CPK体系的扩展

在实际环境中，CPK系统所面临的具体应用环境必然是复杂多样的，在分布性、复杂性、工作模式及通信方式等方面都存在着差异，而CPK的体系结构则能够在单KMC单RMC的体系结构（见图1）的基础上进行扩展以适应不同应用环境的需要。CPK体系的扩展主要包括两类：

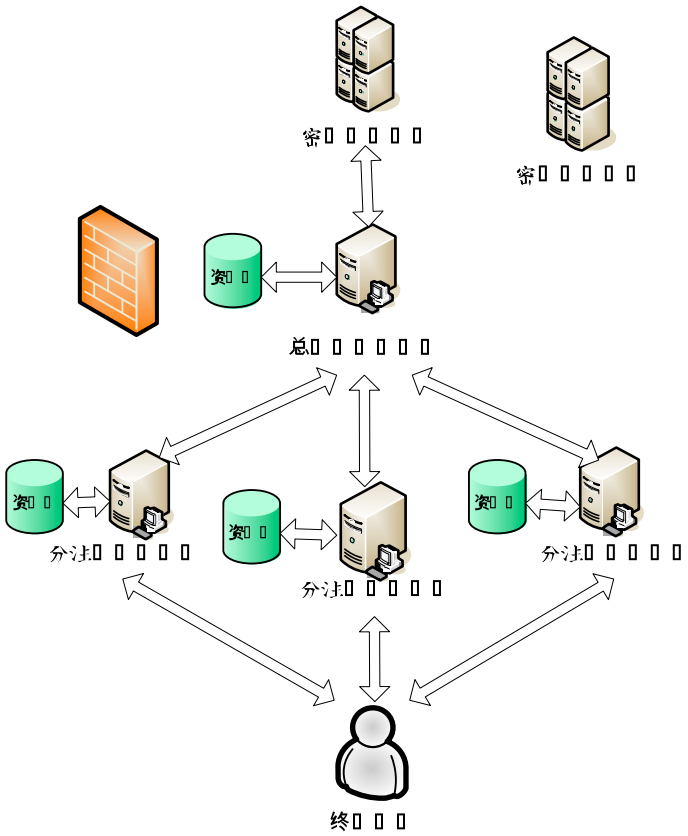
- 采用单KMC多RMC的CPK系统；
- 支持多KMC多RMC的CPK系统。

下面将对这两类体系结构的特点及应用进行详细的分析。

5.1 单KMC多RMC

在CPK系统中，RMC需要以物理方式（面对面）对终端实体进行直接的身份验证，并当面发放ID证书。然而，对规模较大的分布式应用（如电子银行）而言，其业务分布在广域范围内，这就对CPK的用户身份认证机制提出了分布式的需求。因此，为方便终端实体申请ID证书，许多CPK应用系统都需要设置分布式的RMC，来满足实际应用的需要。例如在信用卡系统中，信用卡的终端实体（也即银行客户）分布于全国各个地区，如果只在系统中心（如北京）设立RMC，那么所有要申请信用卡ID证书的终端实体都要到北京的RMC来申请信用卡ID证书，这显然是不切实际。在大规模的分布式应用中，CPK可以在原有的单KMC单RMC的体系结构基

础上扩展为单KMC多RMC的CPK系统。扩展后的体系结构如图所示：



上图描述了单KMC多RMC的CPK体系的基本结构，在实际环境中为适应各种分布式应用的不同需求，该体系结构还可根据资料库的部署机制划分成以下三种类型：

- 轻分注册管理中心
- 普通分注册管理中心
- 强分注册管理中心。

这样划分的依据是在实际的应用系统中，有的分RMC仅要求能够支持终端实体的注册申请和作废申请，而无需对终端实体的相关资料进行管理；有的分RMC则可能需要管理少量的用户信息，但无需对外提供接口；而有的分注册管理中心则要求自治能力比较强，能够处理本分RMC的终端实体的相关信息甚至提供关于作废列表等等的服务。

下面将分别针对不同体系结构的特点及具体实现要求对其进行介绍，主要是从以下几方面考察各体系结构的特点：

- 名字空间的划分
- 分RMC与总RMC的职能划分
- 分RMC与资料库的交互方式
- 资料库数据的设计
- 资料库一致性的维护
- 分RMC和总RMC的通信
- 对内对外接口的设计

5.1.1 轻分注册管理中心

轻分注册管理中心类型只需设立一个总资料库，分RMC不再设立本地的分资料库，资料

库数据的设计参考本节资料库部分的内容。在一些应用中，可能分RMC仅仅需要接收终端实体的注册申请和作废申请，而不需要存储终端实体的相关信息。比如一个规模较小的信用社，它的所有终端实体的相关信息很可能只存放在总行，而分行只负责接收终端实体的注册申请和审核，并把信息传给总行。在这种情况下，可以采用轻分RMC。这种模式对各个分RMC的要求较低，不需要相应的存储设备，无须进行数据维护，在业务的规模较大，需要设立较多的分RMC，但同时希望管理职能相对集中于应用的总中心，或分中心不能承担较复杂的技术及管理职能的情况下，采用这个方案可以大大降低成本和管理的复杂性。

标识空间的划分

标识空间指的一个用户标识的定义范围。划分标识空间是把一个大的标识空间按照一定的规则划分成各个独立的分标识空间。这样做的目的是为了减少标识空间管理的复杂性，把一个大的问题分而治之，便于由几个机构分别管理用户标识。

对不同的应用系统而言，其标识空间的划分规则和方法也不同。例如：

(1) 银行账号

把账号的前若干位设为各个分注册管理中心的标识。这样在每一个分RMC就能够保证用户标识的唯一性，而无须与其它分RMC进行交互。通常情况下，作为用户标识的银行帐号是由银行指定的，而不是由终端实体指定的。

(2) 固定电话号码：

电话号码具有地域性的标识，电话号码前几位为其地域标识——区号。通常情况下，作为用户标识的固定电话号码是由终端实体指定的。

(3) 手机号码：

手机号码的前几位代表了不同的移动公司。通常情况下，作为用户标识的手机号码也是由终端实体指定的。

(4) Email地址

可以通过设置域名，进行用户标识的划分。比如，北京大学信息安全实验室可以通过infosec.pku.edu.cn,与其他的实验室，比如网络net.pku.edu.cn进行区分。

(5) 自然名称

对于某些需要采用用户的自然名称的系统，除了需要用户的名字之外，还可以按照单位名称划分名字空间，确保用户标识的唯一性。

在实际系统中，是否需要划分名字空间与应用的具体需求有关。不过通常情况下，建议划分标识空间，这样可以简化应用系统。下面的小节如果没有特别的说明，默认应用系统已经划分了用户标识空间。

分RMC与总RMC的职能划分

在单KMC多RMC的CPK体系中，RMC分为总RMC和分RMC，相互协作完成基本体系结构中RMC的功能，具体包括接收终端实体的注册申请，对注册申请进行审核，向资料库申请检查标识有效性，向KMC申请证书，通知资料库将标识写入数据库，生成认证卡，将认证卡发放给终端实体以及接收作废申请，作废ID证书。在各自具体职能上总RMC和分RMC有所不同。有的功能必须放在分RMC，有的则必须放在总RMC，有的既可放在分RMC又可放在总RMC。必须放在分RMC的功能包括，接收终端实体的注册申请，对注册申请进行审核，将认证卡发放给终端实体，接收作废申请这个四功能，因为如果把这部分的功能放在总RMC则无法满足应用对用户的分布式注册与认证的需求。必须放在总RMC的功能包括，向KMC申请证书。而向资料库申请检查标识有效性，通知资料库将标识写入数据库，生成认证卡，作废ID证书这四个功能则既可放在总RMC也可放在分RMC。功能放置的位置不同对系统性能的影响也不同，因此需要对其专门论述。

- 向资料库申请检查标识有效性

  将此功能放在分RMC：需要分RMC直接和资料库进行交互，具体的交互见本节分RMC与资料库的交互方式部分。如果总RMC没有对用户标识的名字空间进行划分，分RMC需要搜索整个标识空间，此时资料库的效率就比较低。如果总RMC已经划分了名字空间，则只需要搜索该分RMC的名字空间，资料库的效率会得到比较大的提高，有利于缓解资料库的瓶颈问题。因此，如果把功能放在分RMC，建议划分用户标识的名字空间。

  将此功能放在总RMC：对分RMC和资料库的交互方式没有要求。分RMC在接收注册申请之后，把申请的用户标识转发给总RMC，由总RMC向资料库申请检查标识有效性，然后把结果返回分RMC。在这里如果总RMC划分了名字空间，则需要把用户标识和分RMC的编号传给资料库，才能够较快进行检查。如果没有划分则只需要传入用户标识。前者资料库的效率高于后者。把功能放在总RMC对总RMC的性能要求较高，如果分RMC的请求很多，且要求响应时间较短时，常常会成为瓶颈。
- 通知资料库将标识写入数据库

  此功能的划分应与标识有效性检查的方式相对应，如果检查标识有效性的功能放在分RMC（总RMC），通知资料库将标识写入数据库的功能也应放在分RMC（总RMC），因为分RMC（总RMC）一旦确定标识有效之后就可通知资料库将标识写入数据库。
- 生成认证卡

  在讨论这个功能位置的设置时，要结合认证卡产生的方式，认证卡产生的方式具体可以分成以下三种：

  (1) 认证卡批量生产方式

  在密钥管理中心采用批量方式生产ID证书的情况下，认证卡也由注册管理中心批量生产。在某些特定应用中，终端实体的标识是由KMC按照一定规则自动生成的，这些规则保证了生成的标识在整个系统中是唯一的。例如在信用卡应用中，信用卡的号码由银行指定而不是由用户指定，银行方面只要保证信用卡号码不重复即可。当CPK应用于信用卡业务时，可考虑采用信用卡号码作为标识，并采用自动批量生产方式。KMC可以预先生成大量的证书，并发布到注册管理中心，其中的标识由KMC指定。在这种生产方式下，完全可以由总注册管理中心批量制作卡，然后把卡下发的各个分RMC，这时无需检查用户标识有效性，直接由分RMC指定用户标识给终端实体，然后由分RMC或总RMC把标识写入有效名字库。

  (2) 认证卡请求-应答生产方式

  在密钥管理中心采用请求-应答方式生成ID证书的情况下，注册管理中心实时地处理用户的注册申请，每接到（KMC发来的）一个ID证书，就立刻生成认证卡。这是因为一方面CPK系统中终端实体的标识及其他相关信息需要由终端实体指定，例如在邮件的安全应用中，终端实体在加入CPK系统之前已经拥有邮件帐号，并需要将这个邮件帐号作为自己在CPK系统中的标识。由于无法预知终端实体的邮件帐号，KMC必须在接收到RMC的证书申请之后才能根据确定终端实体的标识是什么，并根据这个标识来生成证书。如果CPK系统使用一些已有的标识，如电子邮件帐号，电话号码、身份证号码、用户的真实姓名等，都需要使用请求-应答方式。另外一方面，由于需要由各分RMC将ID证书写入认证卡并分发给用户，需要各分RMC都配备制卡设备以及相关的技术人员，并要求维持与KMC之间的网络实时安全通信，所以对成本的要求也较高。
- 作废ID证书

  由分RMC写入和由总RMC写入的异同点可以参阅本章5.2.1节。

分RMC与资料库的交互方式

分RMC与资料库的交互方式分为直接交互和间接交互两种：

- **间接交互**是指分RMC与资料库的交互通过总RMC进行中转，资料库由总RMC单独控制。写资料库时，分RMC把终端实体相关信息先传给总RMC，然后由总RMC写入总资料库。读取时，先把读取命令传给总RMC，然后由它再去读取总资料库，最后返回结果。这种方式对总RMC的性能要求比较高，因为它必须处理每一个分RMC的读写请求，同时还要处理每一个分RMC的注册申请和作废申请。这样总RMC很容易成为整个CPK系统的瓶颈。但是总资料库的部署比较简单，不需要对多个RMC进行认证，权限的管理也比较简单，而且几乎可以不考虑同步方面的问题。
- **直接交互**是指分RMC与总RMC共享一个资料库，分RMC直接读写总资料库，而不需要通过总RMC进行中转。在这种方式下，总RMC只需要处理分RMC的注册申请。分RMC的作废申请可以由它自己单独出处理，也可以交给总RMC处理。但是总资料库的部署会比较难，因为需要资料库对各个RMC进行认证，同时还需要设计相应的表格对各个注册管理中心进行权限方面的管理，以及数据方面的同步限制，比如要避免不同的分注册管理中心申请同一个用户标识的情况出现。

以上两种存储终端实体信息的方式，在具体实现中可以根据的实际应用的需要权衡利弊，自行选取。

资料库数据的设计

在轻分RMC模式下只有一个总资料库，总资料库的数据设计大致上与3.4.2节相同。但是如果分RMC与资料库的是直接交互，则这个资料库要增加相应的表格：一个是权限控制表，一个是认证表。

认证表

认证表的数据结构同终端实体表（见3.4.2小节），只是这个认证表的终端实体都是注册管理中心。

权限控制表

1. 注册管理中心编号：即是认证表的终端实体编号
2. 权限：对各部分数据的读写权限的设置

资料库一致性的维护

在轻分RMC模式下的只有一个资料库，所以无需对一致性进行维护。

分RMC与总RMC的通信

分RMC与总RMC通信的具体内容与分RMC与资料库的交互方式及分RMC和总RMC的职能划分密切相关。

- 间接交互

当终端实体提出注册申请时，分RMC审核后需要把终端实体的相关信息，包括用户表和终端实体表的终端实体的编号和用户标识传给总RMC，然后由总RMC检查标识的有效性，并且向KMC提出注册申请，成功后返回成功标识或者ID证书。在这里是否返回ID证书是与ID证书是否由总RMC生产有关，如果ID证书由分RMC生产，则无需返回ID证书，否则返回。

当终端实体提出作废申请时，分RMC需要把用户标识传给总RMC，由总RMC向资料库提出申请更新资料库。

在间接交互的情况下，一方面要求总RMC要有较高的数据处理能力，因为它要处理各个分RMC的读写请求，另一方面，对分RMC与总RMC的带宽要求较高，因为根据前面的分析可以知道它们之间的通信量很大。特别是如果各个分RMC的请求都集中在同一个较短的时段，会造成网络的拥塞，以及总RMC无法及时处理分RMC的请求。这个时候，总RMC就成了系统的瓶颈。解决因为网络拥塞而造成的瓶颈问题的一个可行的方案是在分RMC设立一个终端实体信

息的缓存区，在网络繁忙的时候，先把终端实体的其它信息先放在本地的缓存区，只把用户标识提交给总RMC，这样可以很大程度的缓解网络拥塞。

- 直接交互

直接交互的情况需要根据标识空间是否划分分别进行讨论。通常的情况下，建议对分RMC负责的用户标识划分标识空间，这样有利于管理。但是也有可能出现用户标识是无序的，无法进行标识空间划分的极端情况。

划分标识空间

当终端实体提出注册申请时，分RMC审核后，再检查用户标识的有效性，需要把终端实体的相关信息，包括用户表和终端实体表的终端实体的编号和用户标识写入数据库，然后把用户标识发给总RMC，并把注册申请发给总RMC，由总RMC向KMC转发注册申请，成功后返回成功标识或者ID证书。

作废的时候，可以是由各分RMC或者总RMC修改用户标识的状态。

不划分标识空间：

在这里，大部分的通信情况和划分标识空间的情况是一样的，但需要额外考虑用户标识有效性的检查该如何完成。如果由分RMC完成，则需要多设一个用户标识的状态，已申请状态，这样可避免不同的分RMC申请同一个用户标识。如果由总RMC完成，则由总RMC判断。具体判断的方法是，总RMC维护一个申请列表，如果其它分RMC申请的用户标识已经在列表中存在，则向分RMC返回用户标识无效。

在直接交互的情况下，对总RMC的数据处理能力要求较低，但是对资料库的设计要求较高，要求考虑到认证和权限的要求。它同样也可能存在网络拥塞的问题。资料库也有可能成为瓶颈。本节前面部分提到的解决拥塞的方法同样可以适用于这种情况。

对内对外接口的设计

这一部分的内容以后考虑

5.1.2 普通分注册管理中心

普通分RMC类型在各个分RMC和总RMC都设有资料库。但是各个资料库之间是相互独立的。在一些应用中，分RMC可能除了基本的功能外，还需要负责提供相关的查询服务给终端实体，或者需要给该分RMC的管理者提供相应的统计数据，这个时候就要求分RMC设有本地资料库。

分RMC和总RMC的职能划分

RMC的各自职能与5.1.1轻分RMC部分基本类似，但亦有不同之处。

分RMC必须具有的功能：除接收终端实体的注册申请，对注册申请进行审核，将认证卡发放给终端实体，接收作废申请这个四功能外，还必须提供通知资料库将标识写入数据库这个功能，由于分RMC的资料库存有用户终端实体表，所以必须负责通知资料库将标识写入数据库。另外，为了维护资料库的一致性还需要具有修改用户标识状态的功能，具体的内容见本节资料库一致性维护部分。

总RMC必须具有的功能：向KMC申请证书，修改有效用户标识库，修改作废用户标识库。

分RMC和总RMC协同完成的功能：作废ID证书。

此外，向资料库申请检查标识有效性和生成认证卡这两项功能，既可以放在分RMC实现也可以放在总RMC实现。

如果把向资料库申请检查标识有效性功能放在总RMC，则总RMC要检查有效用户标识库和作废用户标识库两个部分，效率较低，而且也会增加总RMC的负担，有可能使RMC成为瓶颈。所以建议把这个功能放在分RMC，在这种情况就要求应用系统一定要划分用户标识空间，否

则分RMC需要与其它的分RMC进行通信来确定用户标识的唯一性了。

生成认证卡这个功能可根据应用决定是放在分RMC还是放在总RMC。在一些应用中，可能要求终端实体在提出注册申请之后的很短时间内就能够拿到认证卡，这个时候最好把生成认证卡的功能放在分注册管理中心。而其它的情况则可以根据成本的高低和管理的复杂性以及安全性的限制自行选择该功能设置的位置。如果放在分RMC，则要求各个分中心都要配备相应的生产设备，前期投资的成本比较高，而且因为各个分RMC都有生产认证卡的权利，容易在系统人员内部出现安全漏洞。如果放在总RMC，要求定期的把生成的卡分发给各个分中心，管理比较复杂，长期的成本较高，但是可以把系统内部人员造成的安全性的漏洞控制在一个较小的范围。

资料库数据的设计

普通分RMC类型的资料库分成分RMC的资料库和总RMC的资料库两个部分:

分RMC的资料库包括用户表和终端实体表。

总RMC的资料库包括：有效用户标识和作废用户标识，安全网络表，公钥因子矩阵。

这种资料库的设计方案带来了一个新的问题，那就是如何维护分RMC终端实体表的用户标识的状态与总RMC的有效用户标识和作废用户标识的一致性问题。具体的解决方案如下。

资料库一致性的维护

普通分RMC类型的资料库一致性问题主要是针对用户标识的状态而言的。在应用系统已经了划分了用户标识空间的情况下：

当终端实体向分RMC提出注册申请时，终端实体审核后，先检查用户标识的有效性，如果有效向本地的资料库的写入用户标识，并把它的状态设为申请。同时向总RMC提出注册申请，总RMC把命令转发给KMC，等到总RMC收到ID证书，它把该用户标识写入有效用户标识库，同时向分RMC返回成功标志，分RMC收到后，相应的把在本地资料库的用户标识状态改为有效。在操作的过程中，如果一定时间内没有收到成功标志或收到失败标志，则分RMC应该重新提出注册申请。

当终端实体向分RMC提出作废申请时，终端实体首先修改本地的资料库，把用户标识状态改为无效。然后把作废请求发给总RMC，总RMC把用户标识的写入作废用户标识库，再把该用户标识从有效用户标识库中删除，成功后向分RMC返回成功标志。如果分RMC在一定的时间内没有收到返回的成功信息或者收到操作失败的信息，应该重发该请求，直至成功。

分RMC和总RMC的通信

在普通分RMC类型中，通信比5.1.1节要简单。

当终端实体提出注册申请时，分RMC审核后先检查用户标识的有效性，然后把它传给总RMC，总RMC向KMC提出注册申请，成功后返回成功标志或者ID证书。

当终端实体提出作废申请时，分RMC需要把用户标识传给总RMC，总RMC进行相应的处理然后返回成功标志。

其它更具体的交互过程则涉及到资料库，可以参考资料库的一致性维护部分。

对内对外接口的设计

考虑给管理者提供的接口，以及对外提供的接口，这一部分的内容与下面要讨论的强分RMC有很大的区别。是论述的重点之一。讨论的细节有待补充。

5.1.3 强分注册管理中心

强分RMC类型采用分布式资料库，资料库分布在各个分RMC以及总RMC。强分RMC类型的各个分RMC的自治性较强，而且与其它分RMC的交互能力也较强，适合那些对安全性要求较高，终端实体进行通信时需要检查用户标识有效性的系统。这种类型能够在不采用额外的系统比如LDAP，就能够较好的对外提供服务。所以该类型的强项在于对外提供服务。

分RMC和总RMC的职能划分

与5.1.1、5.1.2相比，在此各RMC的职能划分的不同之处在于：

在不考虑对外接口的情况下，总RMC的职能非常的简单，它只需要负责向KMC申请证书以及划分标识空间。

在考虑对外接口的情况下，总RMC要处理从一个分RMC到另外一个分RMC的通信问题。比如，在分RMC（简称A）的终端实体要检查在分RMC（简称B）的终端实体的用户标识的有效性，这个时候A需要把检查请求通过总RMC转给分RMC，结果的传递则从B传给总RMC，然后返回A。

分RMC具备其它所有的RMC功能，除了生产认证卡可选外。如何决定把生产认证卡功能的放在总RMC还是分RMC，可以参考5.1.2节。

资料库数据的设计

强分RMC类型的资料库也分成分RMC的资料库和总RMC的资料库两个部分:

分RMC的资料库包括用户表和终端实体表，公钥因子矩阵，安全网络表和有效用户标识表，以及作废标识表。

总RMC的资料库包括公钥因子矩阵，安全网络表，分RMC的相关信息表以及标识空间划分表。分RMC的相关信息表与终端实体表信息基本一致，除了把终端实体的编号换成分RMC的编号。标识空间表包括分RMC的编号以及它所管辖的标识空间的编号，划分标识空间的方法参见5.1.1节。

在这个设计方案中，各个资料库中都有公钥因子矩阵，安全网络表，所以应该保证这些数据的一致性，一致性维护的细节如下。

资料库一致性的维护

当CPK系统进行更新时，比如更换公钥因子矩阵，或者安全网络表等等，则首先更新总RMC的资料库中的公钥因子矩阵和安全网络表。因此，分RMC的资料库中的公钥因子矩阵和安全网络表的内容以总RMC的为依据。维护分RMC与总RMC的一致性的方法有两种，一种是推方式，另一种则是拉方式。推方式是指每当CPK系统进行更新的时候，由总RMC通知各个分RMC进行更新。拉方式是指分RMC定期的询问总RMC，公钥因子矩阵和安全网络是否和总RMC的一致，如果不是则从总RMC取出最新的信息，然后更新分RMC的资料库。在此建议采用推方式，这是因为CPK系统更新不会很频繁，每次更新，通知每一个分RMC，能够保持很好的一致性，而且占用的资源也很少。如果采用拉方式定义查询的间隔时间比较困难，间隔时间太短，会占用较多的带宽资源，间隔时间太长，一旦出现更新，对终端实体提供的对外接口影响较大。

分RMC和总RMC的通信

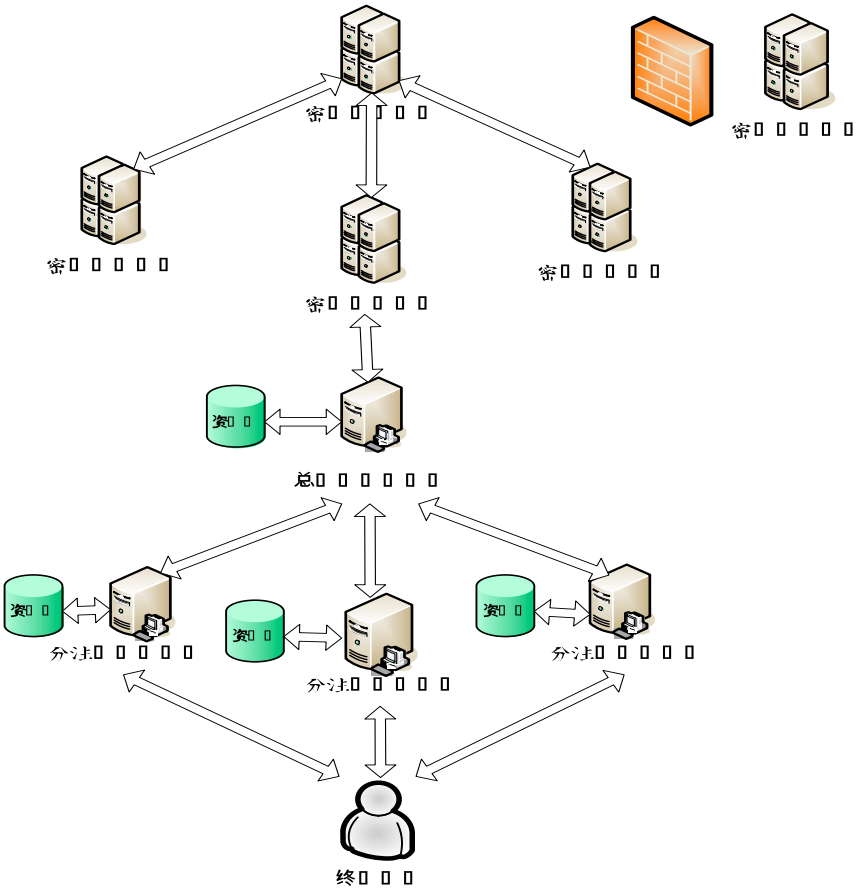
在不考虑对外提供的接口，则它们之间的通信可以参考5.1.2节。

对内对外接口的设计

待补充。

5.2 多KMC多RMC

在一个实际的CPK系统中，除了有分布式的要求之外，系统的规模也是一个很重要的考虑因素。单KMC多RMC系统可以较好的满足分布式的要求，但是它适合的应用规模较小，这主要是因为一个KMC生成证书的能力要受到硬件软件等方面的限制，不能超过某一个上限，可扩展性比较差。而且即使KMC生成证书的能力没有限制，KMC和总RMC，总RMC和分RMC之间的通信量也会成为制约系统性能的一个重要因素。此外，如果一个CPK系统的ID证书只由一个KMC生成，一旦发生私钥因子矩阵泄露或者CPK系统更换新的私钥因子矩阵，波及的范围会很大，原来所有终端实体的认证卡都需要更新，成本会很高。一个典型的例子是一个大型的银行机构，它的业务范围分布全国各地，如果只设立一个KMC，KMC会因为超负荷，成为瓶颈。而且一旦发生私钥因子矩阵泄露或者CPK系统更换新的私钥因子矩阵，则全国所有的终端实体的认证卡将要更换，这显然不合适。因此，需要进一步为各个地区设立分KMC。正是因为存在对扩展性和安全性的进一步的需求，CPK系统引入多KMC多RMC的体系结构（见下图）。



在多KMC多RMC的体系结构中，除也需考虑5.1节中涉及到的各种问题外，主要还应考察分KMC和总KMC之间的功能划分和公私钥因子矩阵的设置等问题。

5.2.1 分KMC和总KMC之间的职能划分

在多KMC多RMC的体系结构图中，建立了两层（或多层）KMC,高层的KMC称之为总KMC，低层的则称之为分KMC。它们在系统中的职能各不相同。一般来说，总KMC负责管理各个分KMC及它们之间的通信安全。分KMC负责为RMC生成ID证书，以及该分KMC对应的子系统内的安全

通信。有时，也有可能由总KMC负责管理各个分KMC，以及整个系统的安全通信。而分KMC仅仅负责为RMC生成ID证书。后一种划分职能的方法，导致总KMC管理范围太大，各个分KMC的子系统的自治能力较差。因此，建议采用前一种划分职能的方法。

在此依据前一种职能划分的方法，详细阐述总KMC和分KMC的职能。

总KMC的职能

在前面提到，在多KMC多RMC的体系结构中，建议总KMC的职能是管理各个分KMC及与分KMC的通信安全。具体的可以分成应用无关的功能和可选的功能。应用无关的功能是指与实际应用系统无关的部分，是每一个采用这种体系结构的CPK系统都必须实现的。可选的功能是指该部分的功能可以根据实际应用系统的需求进行选取。

应用无关的功能具体包括：

- 初始化总KMC与各分KMC的安全通信信道，结合4.1节的相关内容可以更好地了解这一部分的内容：
  (1) KPC将生成的KPC公钥证书，内部系统参数表，系统参数表通过物理信道发放到总KMC。发放的方法可以是管理员将要发放的数据通过移动硬盘或者其它存储设备拷贝到目的地，也可以将该公钥证书硬编码到总KMC中。
  (2) 总KMC通过接收到的内部参数表生成各个分KMC的证书，并通过安全信道发放给分KMC，可以通过网络，也可以不通过网络，由于此时分KMC没有总KMC的公钥，不能对这些数据进行验证，必须由系统中的管理员来控制证书的分发并对其验证。同时，总KMC也生成自身的ID书。在这种情况下，无论是总KMC还是分KMC都是CPK系统的一个终端实体。它区别于使用这个应用系统的终端实体的地方在于前者利用证书传输系统相关信息，而后者用来传递用户数据，而不是系统相关信息。
  (3) 通过以上两个步骤，初始化完成，总KMC和分KMC之间就可以通过证书进行安全通信。但是在实际的应用中，为了保证系统的安全，分KMC和总KMC的证书往往需要定期的更换。有了以上初始化的工作，定期更换证书可以通过已经建立的网络安全信道发放，而无需通过物理信道或者别的安全信道了。
- 各种信息的管理：需要管理的信息包括应用于分KMC的终端实体的信息，私钥因子矩阵，用于生成私钥因子矩阵的参数，自身的ID证书以及私钥因子矩阵与分KMC的对应关系。其中私钥因子矩阵和自身的ID证书必须保密。保密可以通过用户口令进行加密。
- 生成私钥因子矩阵：系统初始化后，总KMC有一个系统参数表，其中包括了私钥因子矩阵。总KMC根据该私钥因子矩阵，利用一些数学函数生成新的私钥因子矩阵，作为分发给分KMC的私钥因子矩阵。
- 下发私钥因子矩阵：在首次初始化系统的时候，需要先初始化总KMC与各分KMC的安全通信信道，然后把私钥因子矩阵和映射函数通过安全信道下发给分KMC。下发私钥因子矩阵可以用对应分KMC的公钥加密，或者用公钥加密会话秘钥，用会话秘钥加密私钥因子矩阵来达到保密的目的。下发私钥因子矩阵应该考虑的另外一个问题是下发的时机，也即什么时候需要下发私钥因子矩阵。一般来说，分KMC在这两种情况下需要更换私钥因子矩阵：第一种情况是私钥因子矩阵已经到期，这是出于安全性考虑，需要给私钥因子矩阵定有效期，在这种情况下，下发可以是由总KMC定期的进行，也可以由分KMC自己申请后下发；第二种情况是分KMC出现了私钥泄露的情况，需要提前作废私钥因子矩阵，这个时候分KMC需要向总KMC提出更换私钥因子矩阵，总KMC根据要求生成私钥因子矩阵并下发。

可选的功能包括：

- 安全策略的制定和分发：如由总KMC制定公司或机构中不同等级的员工和顾客的权限，然后要求分发给各个分KMC执行。

- 全局黑名单的维护：这与对外接口的设计相关，具体的参见对内对外接口设计部分。
- 私钥因子矩阵相关数据的统计：比如私钥因子矩阵的使用的频率等等。这些有助于总KMC制定分发私钥因子矩阵的策略，比如各个分KMC该拥有的多大的私钥因子矩阵。

在实际的应用系统中，可能不需要以上可选的功能，而需要实现其他功能，此时可根据需求自行添加适当的功能。

分KMC的职能

分KMC负责为RMC生成ID证书，以及该分KMC对应的子系统内的安全通信。在这里同样可以把分KMC的功能分成应用无关的功能和可选的功能两个部分。

应用无关的功能包括：

- 初始化子系统的安全通信信道：这个信道包括分KMC与它管辖下的总RMC之间的信道，总RMC与分RMC之间的信道，以及RMC与资料库之间的信道。它的初始化过程可以参考4.1节。
- 生成ID证书：利用私钥因子矩阵和映射算法算出私钥，并生成私钥ID证书。
- 申请私钥因子矩阵：在CPK系统中，出于安全性的考虑，需要定期的更换私钥因子矩阵。当私钥因子矩阵到期时，分KMC就要向总KMC申请，当然，也可以由总KMC进行定期的更新。但是，一旦分KMC的私钥因子矩阵发生泄漏的情况，分KMC必须及时的向总KMC申请更换。并且要作废由该私钥因子矩阵生成的认证卡。

可选的功能则包括：

- 配置安全策略：可以根据总KMC的安全策略进行配置，也可以根据子系统的需要进行相关的配置。
- 收集私钥因子矩阵相关数据并传给总KMC:收集该子系统中私钥因子矩阵的使用情况，传给给总KMC。

可选功能的实现也需依据应用系统的实际需要而定。

5.2.2 私钥因子矩阵的设置

在多KMC多RMC的体系结构下，CPK系统设置私钥因子矩阵的方式有三种：

- 总KMC和所有的分KMC共享一个私钥因子矩阵。

  在这种情况下，一旦某一个KMC的私钥因子矩阵泄露了，无论是由总KMC还是由分KMC泄露的，整个安全系统的都会崩溃，不仅系统之间的通信信道需要重新配置，所有终端实体的认证卡都必须更换，影响非常的大，而且出现泄露的几率也高，因为该私钥因子矩阵在所有的KMC都有备份，只要有一个KMC的安全防范做的不够就会影响全局。但这种方式的优点则在于由于公用一个私钥因子矩阵，系统中的所有ID证书之间都可以任意相互认证并实现签名及加密业务的互通，所以在具体实现中应根据实际情况在安全性与业务特征之间进行适当的折衷。
- 总KMC有一个单独的私钥因子矩阵，其它所有的分KMC都共享一个私钥因子矩阵。

  在这种情况下，系统之间的通信信道可以独立维护，其他则与上一种情况类似。
- 总KMC和所有的分KMC都分别拥有不同的私钥因子矩阵。

  在这种情况下，只要总KMC的私钥因子矩阵不发生泄露，可以避免发生上面的问题。如果不发生某个分KMC的私钥因子矩阵泄露的事故，它的波及范围只限制它所在的子系统。从安全性能的角度考虑，这种情况是最合适的。但是第三种情况也会给不同子系统终端实体之间的交互带来困难。比如，子系统A的终端实体C要求与子系统B中的终端实体D进行保密通信。这时C只知D的用户标识，由于D和C的公钥因子矩阵与C不同，而且映射

算法也不同，C无法单独与D进行通信，必须借助中间机构，如RMC或者KMC，才能够进行通信。

5.2.3 子系统之间的交互

在总KMC和所有的分KMC都分别拥有不同的私钥因子矩阵情况下，不同子系统的终端实体之间的就无法直接利用该终端实体的公钥因子矩阵直接映射，而需要通过一定的途径获得另外一个终端实体对应的公钥因子矩阵和映射算法。具体的解决方案可分为两种查找要分成两种情况讨论：

- 所有子系统在同一个标识空间：在这种情况下，建议划分标识空间。以信用卡为例，在同一个银行机构的信用卡是唯一的，而且在不同的分行申请的信用卡，信用卡的前几位则不同。划分之后，不同子系统地终端实体间的交互在终端实体看来就像和本子系统的终端实体的交互一样，也就是说，对终端实体来说是透明的。具体的处理方法有以下几种：
  (1) 通过总KMC，维护一个公钥因子矩阵和分KMC的对应表（每个KMC对应一个子系统）。这样，每当子系统A的终端实体C要求与子系统B中的终端实体D进行保密通信，C只要把D的用户标识传给RMC，RMC根据标识得知是B子系统的，通过分KMC向总KMC查询B字系统的公钥因子矩阵和映射算法。
  (2) 在每个子系统的资料库都有一个公钥因子矩阵和分KMC的对应表，而且总KMC也维护一个公钥因子矩阵和分KMC的对应表。这样就没有必要每次不同子系统之间的交互都要访问总KMC了。但是要注意维护各个子系统和总KMC中对应表的一致性。一致性维护的方法，可以由总KMC每次改动后通知各个子系统地资料库更新。也可以是子系统的资料库定时向总KMC提出更新。
  (3) 如果通信还要求知道对方的用户标识是否作废，则还需要维护一个作废列表。这个时候需要设立一个和总KMC在同一个层次上的总RMC,用来管理这些信息。如果没有设立相应的RMC来管理这些信息，则每次请求都需要总KMC把检查用户标识有效性的请求转发到其它的子系统，这样响应速度通常会比较慢。
- 各子系统都有自己的标识空间：比如，不同银行机构之间，它们的用户标识空间各不相同。不同子系统的终端实体间的交互本身即无法做到透明。它需要额外的对子系统进行编号，编号之后，需要建立公钥因子矩阵和编号之间的对应关系。而且终端实体与另一子系统的终端实体进行保密通信时，需要知道另一子系统的编号。其它细节的讨论可以参考本节的所有子系统在同一个标识空间的情况。

三、安全性及性能

私钥因子矩阵的机密性是组合公钥技术安全性的关键。由于对椭圆曲线密码系统来说，仅知道公钥（因子）$r \cdot G$，要求出对应的私钥（因子）r是极其困难的。所以试图通过公钥因子矩阵逆推出对应的私钥因子的攻击方法是不可行的。

另一方面，通过掌握多个用户的私钥，列方程组求解私钥因子矩阵的方法也是不可行的。假设攻击者获得了两个用户的私钥SK_a、SK_b，并假设在与这两个用户相对应的n层映射中，只有其中一层的映射值不同，其他映射值全部相同。那么（$SK_a - SK_b$）可以消除$n-1$个相同的私钥因子的影响，但得到的仍是两个不同的私钥因子之差，而不是私钥因子本身。所以这种方法也无法暴露出私钥因子，私钥因子矩阵仍然是安全的。

私钥因子矩阵需要妥善地保存在密钥管理中心，而公钥因子矩阵则可以在公共媒体中公布，也可以保存在专用媒体中直接分发给用户使用。设组合公钥系统的因子矩阵大小为$m\times n$，那么所需的因子存储量即为$m\times n$，而可以由这些因子组合出的密钥量却为m^n。表1给出了因子矩阵大小和密钥组合量的比较。如：当矩阵大小为（32×16）= 512时，因子矩阵的存储量为512个密钥，而可组合的密钥量则为$(32)^{16}=10^{24}$。而当矩阵大小为（16×64）= 1024时，存储量为1K个密钥，而组合密钥量为10^{77}。

表1. 组合公钥技术的因子矩阵大小与组合密钥量

m	n	因子矩阵大小（密钥存储量）	组合密钥量（实际密钥量）
$32=2^5$	$16=2^4$	$2^9=512$	$(2^5)^{16}=2^{80}\approx 10^{24}$
$64=2^6$	$16=2^4$	$2^{10}=1K$	$(2^6)^{16}=2^{96}\approx 10^{28}$
$128=2^7$	$16=2^4$	$2^{11}=2K$	$(2^7)^{16}=2^{112}\approx 10^{33}$
$256=2^8$	$16=2^4$	$2^{12}=4K$	$(2^8)^{16}=2^{128}\approx 10^{38}$
$32=2^5$	$32=2^5$	$2^{10}=1K$	$(2^5)^{32}=2^{160}\approx 10^{48}$
$64=2^6$	$32=2^5$	$2^{11}=2K$	$(2^6)^{32}=2^{192}\approx 10^{57}$
$128=2^7$	$32=2^5$	$2^{12}=4K$	$(2^7)^{32}=2^{224}\approx 10^{67}$
$256=2^8$	$32=2^5$	$2^{13}=8K$	$(2^8)^{32}=2^{256}\approx 10^{77}$
$16=2^4$	$64=2^6$	$2^{10}=1K$	$(2^4)^{64}=2^{256}\approx 10^{77}$

四、CPK技术特点

简述公钥密码学的发展历史，在密钥管理的框架下论述PKI及CPK的各自特点，分析PKI的应用范围和局限性，对比提出CPK的优点。

PKI没有从技术上解决规模化问题，而是靠增加CA层次的结构化方式解决，而没有解决一个CA的规模化；PKI靠第三方证明的方式解决标识和密钥的一体性，而没有从技术上解决。由此带来一系列难以逾越的矛盾。目前的思路逐步转向CPK认证系统。

五、典型应用系统

1. 电子银行认证系统

2. 邮件认证系统

3. 手机认证系统

4. 证券认证系统