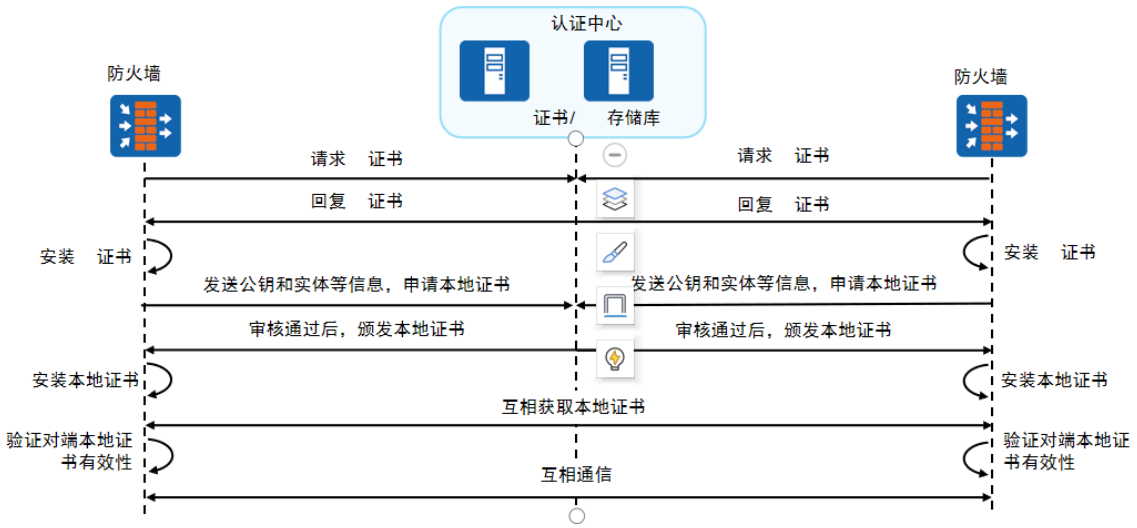
1. PKI/CA 认证体系。

PKI （ Public Key Infrastructure ）指的是公钥基础设施。PKI 从技术上解决了网络通信安全的种种障碍。

CA （ Certificate Authority ）指的是认证中心。CA 从运营、管理、规范、法律、人员等多个角度来解决了网络信任问题。

1.1. 工作机制。



1.2. PKI/CA 组成。

从总体构架来看， 一个 PKI/CA 体系由以下四部分共同组成。

- 终端实体

  他是PKI产品或服务的最终使用者，可以是个人、组织、设备（如路由器、交换机）或计算机中运行的进程。

- 证书认证机构 CA

  它是PKI的信任基础，是一个用于签发并管理数字证书的可信实体。其作用包括：发放证书、规定证书的有效期和通过发布CRL确保必要时可以废除证书。

- 证书注册机构 RA

  RA是CA的延伸，可作为CA的一部分，也可以独立。RA功能包括个人身份审核、CRL管理、密钥对产生和密钥对备份等。PKI国际标准推荐由一个独立的RA来完成注册管理的任务，这样可以增强应用系统的安全性。

- 证书/存储库

  负责证书和CRL的存储、管理、查询等。

1.3. 认证机构 CA 的类型。

- 自签名 CA

  在自签名 CA 中，证书中的公钥和用于验证证书签名的公钥是相同的。

- 从属 CA

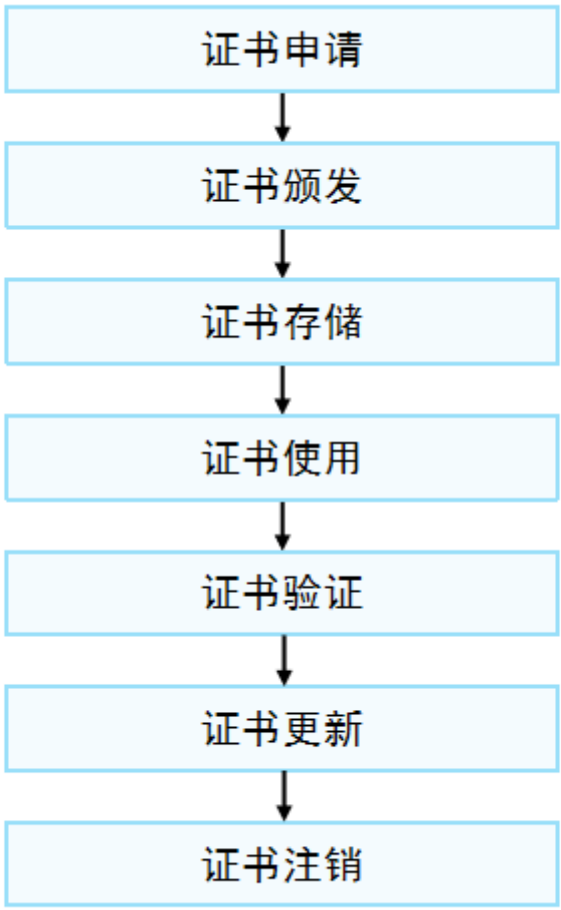
  在从属 CA 中，证书中的公钥和用于验证证书签名的公钥是不同的。

- 根 CA

  根 CA 是一种特殊的 CA，它受到客户无条件地信任，位于证书层次结构的最高层。所有证书链均终止于根 CA 。根 CA 必须对它自己的证书签名，因为在证书层次结构中再也没有更高的认证机构。

1.4. 证书申请流程。

PKI的核心技术的核心技术就围绕着本地证书的申请、颁发、存储、下载、安装、验证、更新和撤销的整个生命周期进行展开。



2. CA 搭建。

1. 创建 CA 目录。

```
$ mkdir myca myca/newcerts myca/private
$ touch myca/index.txt
$ echo '01' > myca/serial
$ cp /etc/pki/tls/openssl.cnf /home/omm
```

2. 拷贝并编辑配置文件 openssl.cnf 。

```
# vi openssl.cnf

[ CA_default ]
dir             = /home/omm/myca           # Where everything is kept
default_md      = sha256
```

3. 生成 CA 私钥。

```
$ openssl genrsa -aes256 -out myca/private/cakey.pem 2048
```

4. 生成根证书请求文件。

```
$ openssl req -config openssl.cnf -new -key myca/private/cakey.pem -out
myca/careq.pem

Country Name (2 letter code) [AU]:CN
State or Province Name (full name) [Some-State]:guangdong
    Locality Name (eg, city) []:guangzhou
Organization Name (eg, company) [Internet Widgits Pty Ltd]:tgg
Organizational Unit Name (eg, section) []:tgg
Common Name (eg, YOUR name) []:opengauss
```

5. 生成自签发根证书。

```
# vi openssl.cnf

basicConstraints=CA:TRUE
```

```
$ openssl ca -config openssl.cnf -out myca/cacert.pem -keyfile
myca/private/cakey.pem -selfsign -infiles myca/careq.pem
```

3. 生成证书。

1. 修改 openssl.cnf 文件 和 index.txt.attr 文件。

```
# vi openssl.cnf

basicConstraints=CA:FALSE

# vi myca/index.txt.attr

unique_subject=no
```

2. 生成证书私钥。

```
$ openssl genrsa -aes256 -out server.key 2048
```

3. 去掉私钥的密码保护

```
$ openssl rsa -in server.key -out server.key
```

4. 生成证书请求文件。

```
$ openssl req -config openssl.cnf -new -key server.key -out server.req
```

5. 生成证书。

```
$ openssl ca -config openssl.cnf -in server.req -out server.crt -days 3650 -
md sha256
```

6. 将密钥转化为DER格式。

```
$ openssl pkcs8 -topk8 -outform DER -in client.key -out client.key.pk8 -
nocrypt
```

4. 证书安装。

1. 修改 postgresql.conf 。

```
# vi postgresql.conf

ssl = on
ssl_ciphers = 'ALL'
require_ssl = on
ssl_cert_file = '/home/omm/server.crt'
ssl_key_file = '/home/omm/server.key'
ssl_ca_file = '/home/omm/myca/cacert.pem'
```

2. 配置客户端参数。

```
export PGSSLCERT="/home/omm/client.crt"
export PGSSLKEY="/home/omm/client.key"
export PGSSLMODE="verify-ca"
export PGSSLROOTCERT="/home/omm/myca/cacert.pem"
```