Introduction

The goal of this document is to describe how certificate authentication works with an AppGate server and how it is configured. Readers who are not already familiar with certificates, public key cryptography and PKI should read the chapters "**A Quick Introduction to Public Key Cryptography**" and "**A Quick Introduction to PKI**" to get sufficient background.

This is a technical document but it does not go into the mathematical details of certificates and public key cryptography. Nor does it explain how to set up and administer a PKI; it is assumed that an X.509 based PKI with a CA that can issue user certificates is already available.

Why Certificates?

Using certificate authentication to log on to AppGate effectively gives certificate authentication for applications that don't support it natively.

Compared to Passwords

Passwords are vulnerable to brute force attacks. Automated attacks scan the entire Internet for servers and blindly try to log on with common user names and passwords. Although it is possible to choose a strong password, the more accounts are available the higher the probability that some of them use passwords that are easy to guess. AppGate servers have a number of built-in countermeasures that mitigate the risk of automated attacks, but it is still important to avoid too simple passwords.

Using certificates for authentication gives several advantages over plain password authentication. The cryptographic keys can't be guessed by brute force contrary to most passwords that are chosen by end users. When a private key is stored in a way that resists copying – for example on a smart card – it can be used for two factor authentication since the private key is "something you have" and the passphrase that is needed to unlock it is "something you know". It is important to protect the private key from copying since someone who has access to a copy can try different passphrases offline.

Compared to SSH Public Keys

Certificates differ from SSH public keys in that the public key does not contain the user's identity. When SSH public keys are used, the AppGate server must know each user's public key in advance. Certificates have the advantage that the server only needs to know about and trust the CA that issues certificates, it can then verify that the certificates presented by the users are genuine.

Compared to One Time Passwords

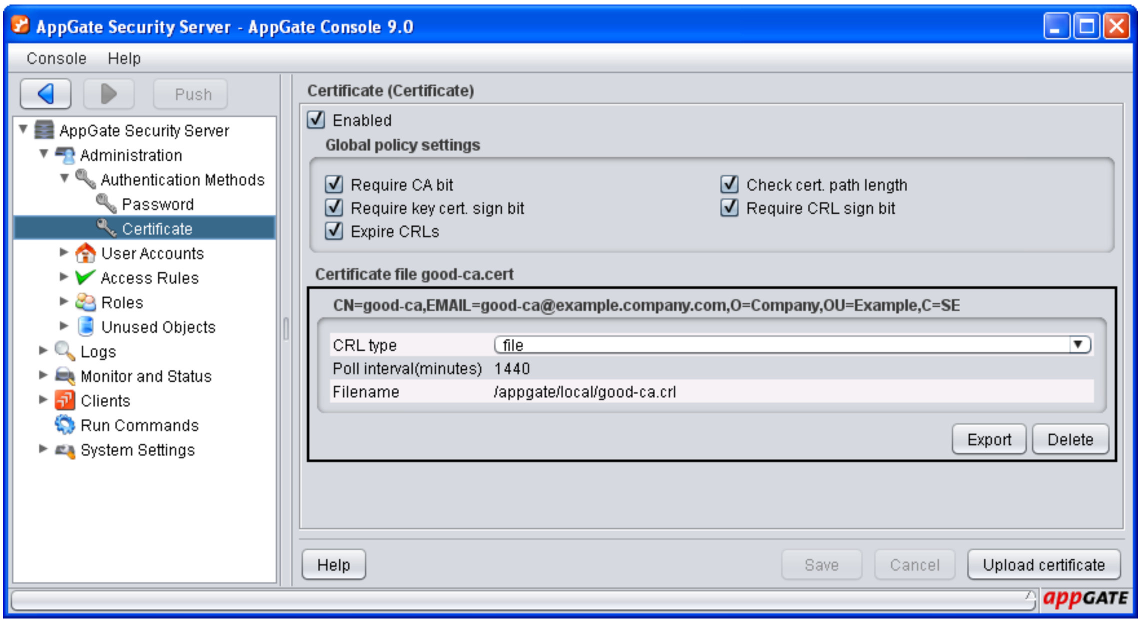
Compared to one time passwords, certificates do have the advantage that an attacker can't brute force certificate authentication online. Although one time passwords can be quite safe if they are long enough, a brute force attack may cause accounts to be locked due to too many failed attempts to log in. Short one time passwords may be less secure than they appear at a first glance. If the one time password consists of a four digit PIN code, an account is locked after five failed attempts and there are 1000 valid users, then an attacker has 40% chance of breaking in to at least one account by trying five times for each user.

AppGate Server Configuration

Setting Up Certificate Authentication

Certificate authentication can be set up in the AppGate server by going to `Administration` → `Authentication Methods` and creating a new certificate authentication method.

Enable certificate authentication and upload the CA certificate. What check boxes should be checked depends on the CA. This is also the place where CRL handling is defined. The AppGate server can fetch CRLs from a web server or an LDAP server. Alternatively the CRLs can be copied to the AppGate server by other means, for example with a cron job.



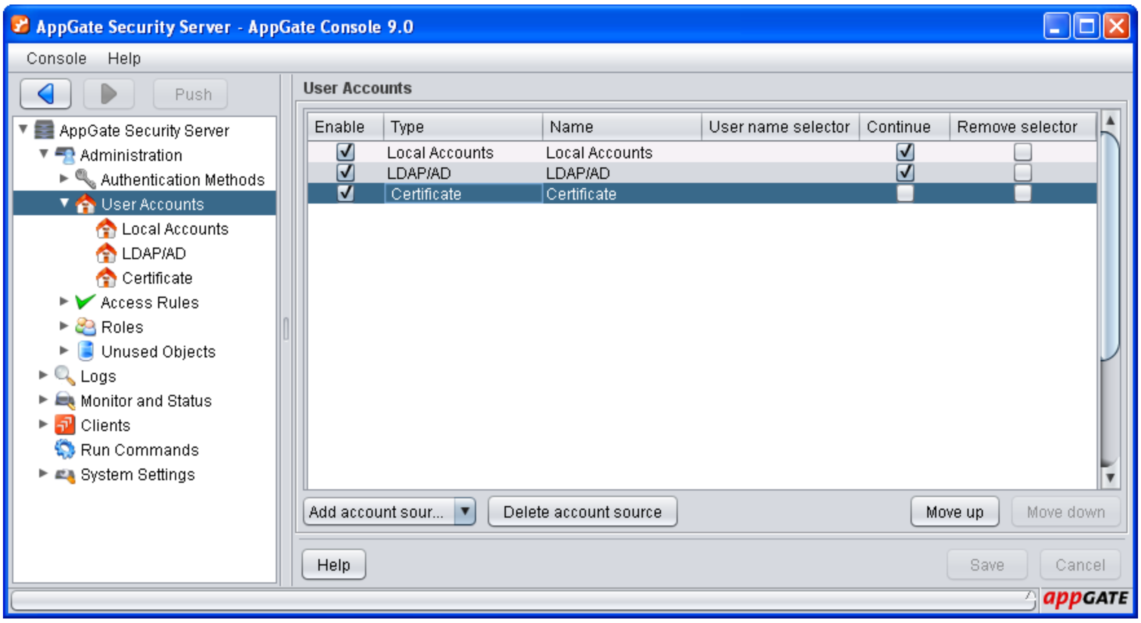
Revocation lists keep track of certificates that are no longer valid. If CRL handling is defined and the AppGate server can't find a valid CRL, for example if the CRL has expired or can't be found in the specified place, then no user will be allowed to log on to the AppGate server with certificates issued by that CA.

Even if a user is authenticated with a valid certificate, access rules defined for roles or services may be configured to block some or all access. Access rules can for example be used as an emergency exclusion of a user if revocation list updates are not frequent enough.

If something doesn't work as expected then check the logs on the AppGate server. The daemon responsible for managing certificates and revocation lists is *ag_pvd*.

Certificate as an Account Source

When a certificate authentication method has been defined it is possible to add a new certificate account source under `Administration` → `User Accounts`.

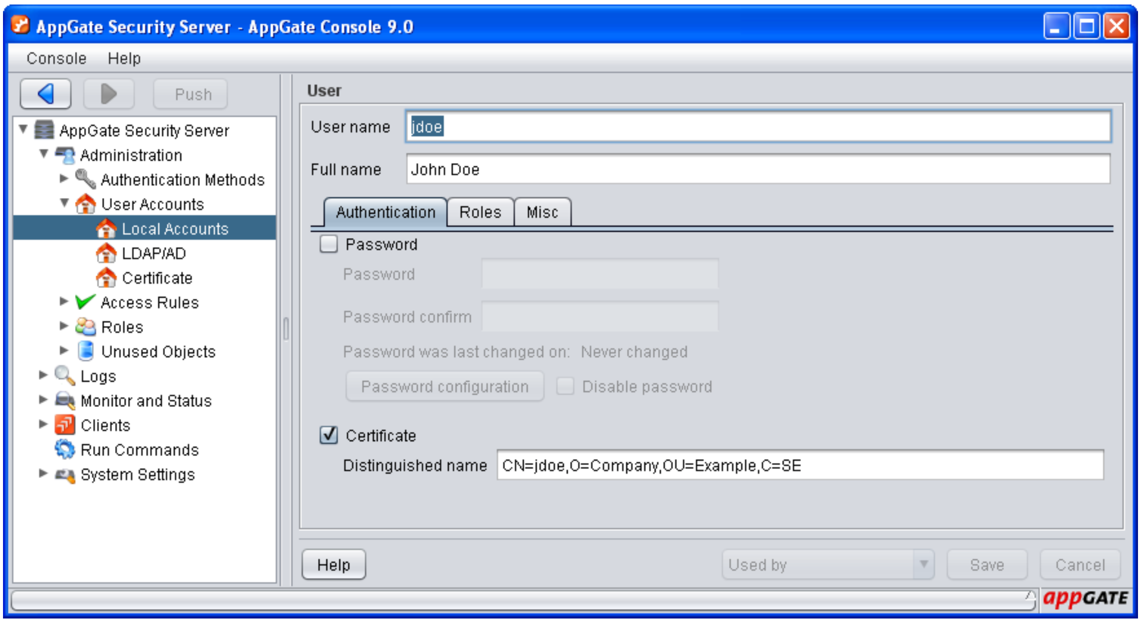


Any user who is successfully authenticated with a certificate will get access to the roles specified for this account source. If different users need different access then that must be defined with access rules. User specific access rules may be complex and error prone to maintain so a better solution is often to use local accounts or LDAP based accounts with certificate authentication as described below.

In order to use certificate authentication with local accounts or LDAP accounts, certificate authentication must be defined but a certificate account source does not need to be defined.

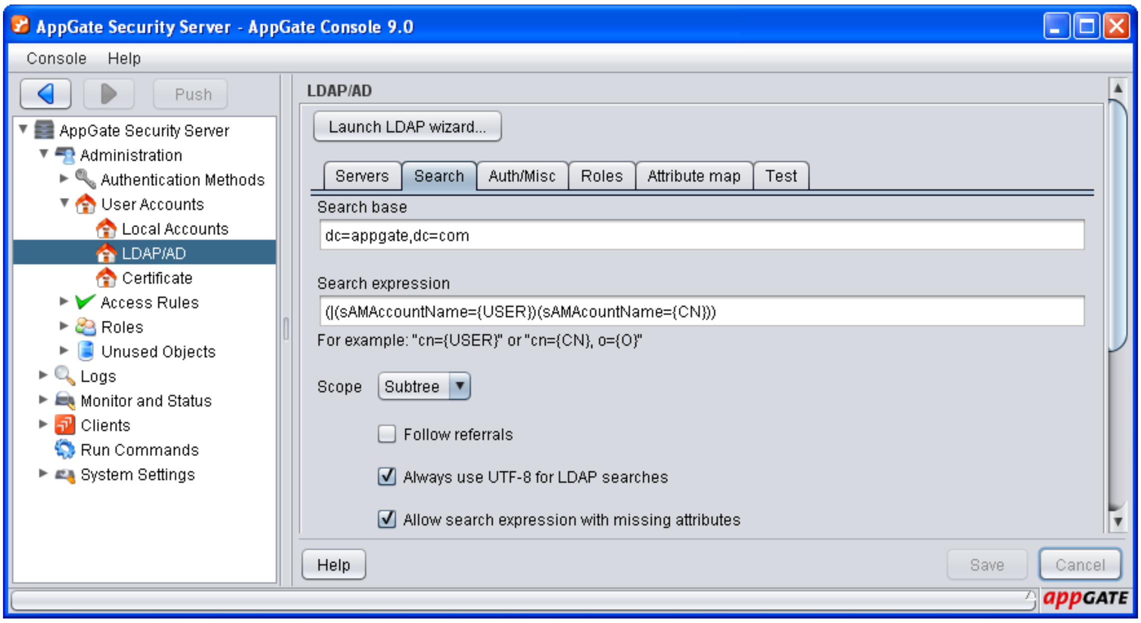
Certificate Authentication with Local Accounts

When a local user has been defined and certificate authentication is configured it is possible to allow the user to log in with a certificate. The definition for the local user should now have a checkbox for “certificate” among the authentication methods and it is possible to fill in the user’s *Distinguished Name* (the subject in the certificate). When this is done the user can log on using the certificate and will get access to the roles defined for this user.



Certificate Authentication with LDAP Accounts

When using LDAP accounts with certificate authentication the LDAP search expression uses parts of the certificate's subject. {USER} will be expanded to the entire subject which is usually not correct. Exactly what the search expression should look like depends on the LDAP server and what the certificate subject looks like. If for example the subject contains the RDN `CN=John Doe` and the LDAP server stores the user's name in the attribute `name` then `name={CN}` could be the search expression to use.



It is also necessary to add certificates to the allowed authentication methods under the "Auth/Misc" tab. When an LDAP user logs on with a certificate access will be granted according to the configuration in the "Roles" tab for the LDAP account source.

Client Configuration

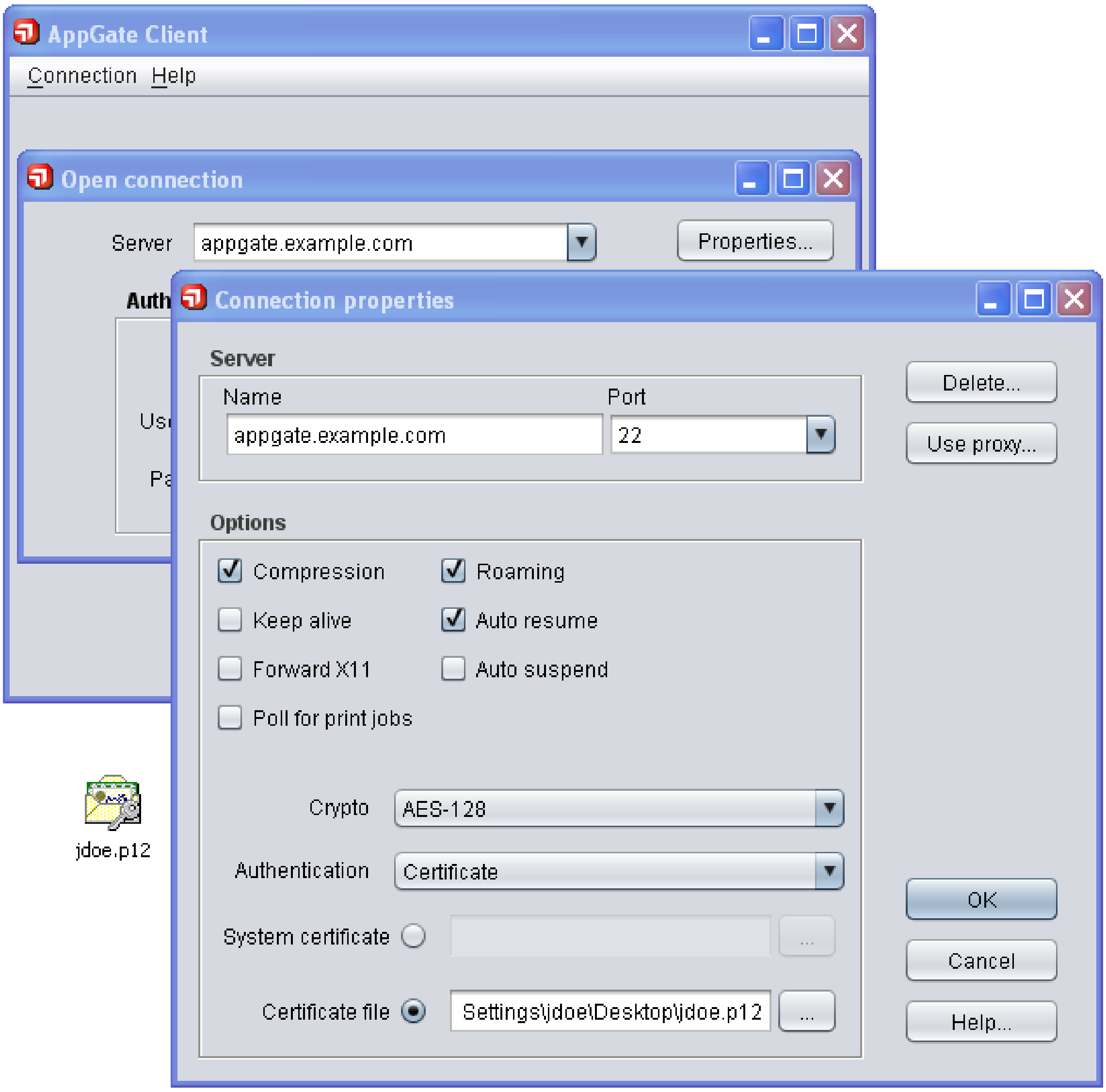
In order to use the certificate authentication method the client must have access to both a valid certificate and the corresponding private key. The certificate and the private key are often referred to as the certificate which strictly speaking is not entirely correct. It is important to protect the private key, but the certificate does not contain any security sensitive information.

Certificates and private keys can be handled in a number of ways. A file containting a certificate and a private key can be stored in PKCS#12 format, which usually contains the private key in encrypted form and requires a password to unlock. Although it is encrypted it should be protected from copying since it is possible to test a large number of passwords in a short time with fast computers.

Another way to manage certificates and keys is to store them on smart cards from which the private key can not be read, but the smart card will perform the required cryptographic operations if the correct password is provided. The smart card manufacturer often provides a library with which the AppGate client can communicate via PKCS#11. For Windows based operating systems the smart cards are usually integrated with MS CAPI which the AppGate client can use.

Workstations

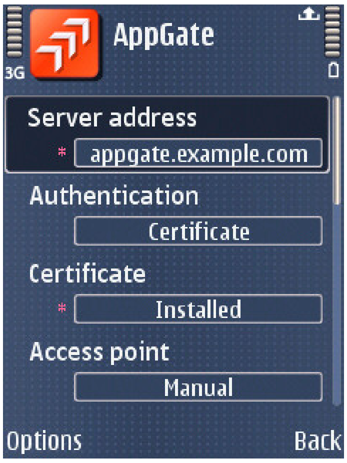
To log on to an AppGate server using certificate authentication from a workstation the authentication method must be set accordingly. When the authentication method for the client is set to certificate it is possible to choose the certificate to use from the *Properties* window.



The user will be prompted for the password to unlock the private key, and if the certificate is valid the user will be logged on.

Mobile Phones

When using certificate authentication from a mobile client the certificate must be copied onto the phone as a PKCS#12 file. When the user chooses certificate authentication a file chooser will appear to allow the user to specify where the PKCS#12 file is located. This only needs to be done once, when the PKCS#12 file is copied onto the phone.



On Symbian based phones the AppGate client will then protect the PKCS#12 file from copying by moving it into an application specific directory. The contents of that directory is not available to other applications, and the PKCS#12 file will be erased if the AppGate client is uninstalled or upgraded.

Windows Mobile does not provide application specific folders that are protected from other applications, but the AppGate client will hide the PKCS#12 file to make it more difficult to find.

There is currently no support for smart card based certificates on mobile phones.

A Quick Introduction to Public Key Cryptography

Public key cryptography is based on mathematic operations that make it possible to create a pair of numbers (the private key K_{priv} and the public key K_{pub}) with particular properties. If Alice generates a key pair she can keep the private key to herself and distribute the public key to others with whom she wants to communicate. The private key must be kept secret but the public key can be freely distributed.

There are a number of different mathematical methods that implement public key cryptography, some examples are *RSA*, *DSA* and *ECC*. These algorithms can be used for either or both of *encryption* and *signing*. Although some algorithms can be used for both encryption and signing it's good practice to not use the same key pair for both operations.

Different algorithms require different key sizes in order to provide the same security. Public key cryptography requires longer keys than symmetric cryptographic algorithms such as AES for the same level of security.

Encryption

Anyone with access to a public key can use it to encrypt a message, and no access to the private key is needed for encryption. The private key is required for decrypting the message.

If Alice has access to K_{priv}, and Bob and Mallory both know K_{pub} then Bob can encrypt a message. Alice can decrypt Bob's message but Mallory is not able to do that without access to K_{priv}.

Signing

The private key can be used to create a signature. Anyone with access to the public key can verify that the signature was created by someone with access to the private key.

If Alice has access to K_{priv}, and Bob and Mallory both know K_{pub} then Alice can sign a message and both Bob and Mallory can verify that the signature was created by Alice.

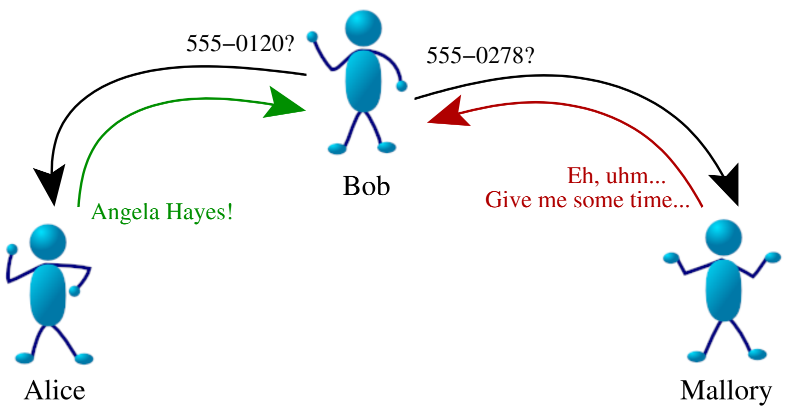
Example

Public key cryptography is based on mathematical operations that require K_{priv} which can be verified without access till K_{priv}. The mathematical details are out of the scope of this document but the principle can be visualised with an example.

Assume that a phone book is used as a public key. The phone book is sorted by name. Bob and Mallory have a copy each of the phone book so given a name they can easily look up the phone number for that person.

Alice has another phone book, the private key. This contains the same names and numbers, but it is sorted on phone number instead. So Alice can do something that Bob and Mallory can't – given a phone number Alice can easily look up whose phone number it is.

With this setup, Alice can prove to Bob that she has access to the private key. Bob can pick a name at random and look up the phone number for that name. Bob then sends the phone number to Alice and asks her to tell him whose number it is. Since Alice has access to the private key she can easily look up the number and give the name to Bob. Mallory has access to the public key but not to the private key and therefore can't pretend to be Alice.



K_{priv}

...	...
555–0119	Dubois, Alison
555–0120	Hayes, Angela
555–0121	Kritschoau, Michael
555–0122	Goodman, Dave
...	...

K_{pub}

...	...
Hawkins, Zeke	555–3126
Hayes, Angela	555–0120
Hayes, Michael	555–0133
Hayworth, Michael	555–0278
...	...

A Quick Introduction to PKI

PKI is short for *Public Key Infrastructure* and is a generic term for a system that manages distribution of public keys. A common PKI architecture is based on X.509v3 certificates which is the topic of this document.

An X.509v3 based PKI can be quite complex. This document contains many simplifications and does not attempt to go into technical details. The information in this chapter is not meant to be sufficient to set up and run a PKI in a secure way, a deeper knowledge of the subject is required to do that.

X.509v3

An X.509 certificate is an association between a public key and an identity. The identity, the public key and some additional information are stored together and this is signed with the private key of the CA that issued the certificate.

Anyone with access to the CA public key can verify the signature of the certificate and, assuming that the CA is trusted, draw the concolusion that the public key does belong to the owner of the certificate whose identity is included in the certificate. The owner can then, by means of public key cryptography, prove knowledge of the corresponding private key, and that in turn means that the private key and certificate can be used for authentication.

A certificate does contain some additional information such as when the certificate is valid and what it can be used for.

The trust in a certificate comes from the fact that it has been signed by a trusted CA.

CA

A CA (Certificate Authority) issues certificates. A key pair must be generated and the CA must verify the identity of the owner before the CA issues a certificate. How the verification is done depends on the design of the PKI and it often requires the user to show up in person and show an ID.

The CA has its own private key which is used to sign the certificates. The CA's public key is included in a CA certificate which is a normal X.509v3 certificate with permission to sign other certificates. The CA certificate can either be self-signed or signed by another CA.

The end systems need a copy of the CA certificate in order to be able to verify certificates issued by the CA.

The trust in self-signed CA certificates comes from the fact that it has been installed beforehand and is stored on the local disk.

CRL

A CRL (Certificate Revocation List) is a list of certificates that are revoked for some reason. The private key corresponding to the public key in a certificate could for example be compromised. When that happens the CA creates a list of all the certificates that have been revoked and signs it. The CRL also contains a validity date.

The CRL is usually officially published and end systems should update their copy of the CRL regularly. If the CRL is not available, has expired or is corrupt then the policy usually requires that no certificates from that CA should be accepted.

The trust in a CRL comes from the fact that it is signed by the same CA that issued the certificates in the list.

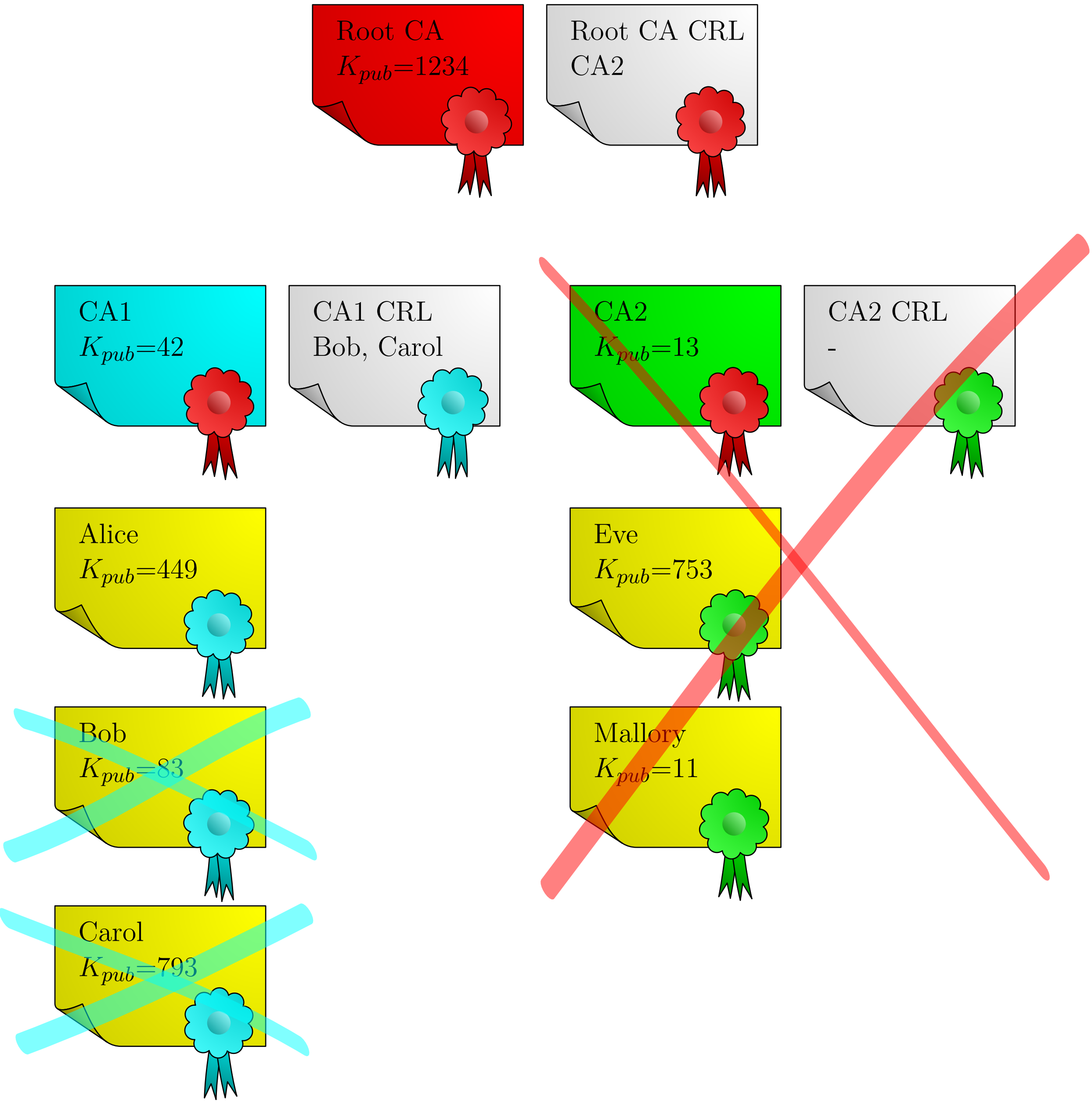
An example PKI

The picture on the next page shows an example of a PKI. There is a self-signed root CA (red) which has issued certificates for two sub-CAs: CA1 (cyan) and CA2 (green). The end user certificates are yellow and the CRLs are gray.

The CRL for the root CA contains CA2 which means that the certificate for CA2 has been revoked and it is invalid together with all certificates issued by it.

CA1 has issued three certificates; to Alice, Bob and Carol. The revocation list for CA1 contains Bob and Carol so their certificates are not valid.

The only valid end user certificate in this picture is Alice's certificate.


Root CA
K_{pub}=1234
Root CA CRL
CA2
CA1
K_{pub}=42
CA1 CRL
Bob, Carol
CA2
K_{pub}=13
CA2 CRL
-
Alice
K_{pub}=449
Eve
K_{pub}=753
Bob
K_{pub}=83
Mallory
K_{pub}=11
Carol
K_{pub}=793

The OpenSSL Command Line Tool

The “openssl” command line tool can be used to examine certificates.

Examine an X.509 Certificate in PEM Format

The following command lists the certificate stored in the file *infile.pem*:

```
openssl x509 -nooout -text -in infile.pem
```

Examine a CRL in PEM Format

The following command shows the CRL and the list of revoked certificates in the file *crl.pem*:

```
openssl crl -noout -text -in crl.pem
```

Examine a CRL in DER Format

The following command shows the CRL and the list of revoked certificates in the file *crl.der*:

```
openssl crl -noout -text -inform der -in crl.der
```

Extracting the Contents of a PKCS#12 file

The following command converts the DER-encoded PKCS#12 file infile.p12 into a PEM file:

```
openssl pkcs12 -in infile.p12 -out outfile.pem
```

Definitions

Asymmetric encryption

Encryption based on public key cryptography. This differs from symmetric encryption in that the encrypter and decrypter use different keys, while symmetric encryption requires that both parties use the same key.

Certificate

A certificate is an association between an identity and a public key. A certificate usually contains other data such as validity time and information on what the key may be used for. Note that the private key is not included in a certificate even though the word is sometimes used sloppily for the combination of a certificate and a private key for example on a smart card or in a PKCS#12 file.

Digital Signature

A signature generated with a private key. This signature can be verified by anyone with access to the corresponding public key.

DSA

Digital signature algorithm (*DSA*) is an asymmetric algorithm that can only be used for signing.

Key pair

A public key and a private key that belong together.

OCSP

Online certificate status protocol (*OCSP*) is an alternative to revocation lists. Instead of using a list containing all revoked certificates, with OSCP the end system contacts an OCSP responder on-line each time it needs to verify a certificate. The responder informs the end system whether the certificate has been revoked or not.

PEM

A PEM encoded certificate is a base-64 encoded DER certificate. This is a text format that allows a certificate to be sent over non-binary channels such as in e-mail.

PKCS#11

An API for communication with smart card based certificates.

PKCS#12

A file format for a DER-encoded certificate and the corresponding private key. The PKCS#12 file is often protected with a password.

Private key

The secret part of a key pair used in asymmetric cryptographic algorithms. The private key should be protected and must not be available to anyone else.

Public key

The public part of a key pair used in asymmetric cryptographic algorithms. The public key does not need to be protected and must be available to all parties with whom the owner of the private key wishes to communicate.

Public Key Infrastructure (*PKI*)

A PKI is used to manage public keys so that a public key is available when needed and it can be verified to whom it belongs. A PKI is often designed around X.509v3 certificates.

RSA

An asymmetric algorithm which can be used for both encryption and signing. The letters RSA are the initials of Rivest, Shamir and Adleman who first described the algoritm.

Secret key

A key used for symmetric cryptography. This key must be known by both parties that communicate and it is usually randomly generated each time a key is needed.

Signature

See *Digital Signature*

X.509v3

X.509v3 is a common format for certificates.