

# CS 415 Project Report: Creating Signed Certificates

## Introduction:

In this report, we present a comprehensive overview of the process involved in creating and managing digital certificates using OpenSSL, a robust and widely-used cryptography toolkit. The objective of this project, as part of CS 415, is to demonstrate a deep understanding of digital certificate creation, configuration, and verification, highlighting the intricate details that ensure secure communications in digital environments.

Certificates play a pivotal role in establishing secure connections over the internet, particularly in verifying identities and encrypting data. The project involves two key stages: firstly, creating a root self-signed certificate, which serves as the cornerstone of trust in a certificate hierarchy; and secondly, using this root certificate to sign a child certificate, personalized with specific user details. This process mirrors the real-world scenario of how certificate authorities and subordinate certificates operate within the digital security landscape.

Throughout the report, we delve into the technical specifics of each step involved in this process. We begin by outlining the configuration and generation of the root certificate, emphasizing the importance of each parameter in the `ca.cnf` file and how it contributes to the certificate's attributes and security features. We then proceed to the creation of the child certificate, detailing how it inherits trust from the root certificate while also being customized to include specific user information and domain settings.










Further, we illustrate the verification process, crucial for ensuring the integrity and correctness of the generated certificates. This step confirms that the certificates contain the correct information and are properly signed, making them ready for deployment in real-world applications.

In the pursuit of excellence and a deeper understanding, the report also includes an extra credit section. This section explores advanced certificate extensions, such as Subject Alternative Names, Key Usage, Extended Key Usage, Basic Constraints, and Certificate Policies. These extensions are not only critical for the nuanced functionality of certificates but also for understanding the diverse applications of certificates in various security contexts.

Through this detailed exploration, the report aims not only to fulfill the project requirements but also to provide a thorough understanding of the practical and theoretical aspects of digital certificate management, a skill set increasingly vital in the modern digital landscape.

## Procedure:

- **Creating a Root Self-Signed Certificate**
  - All Files shown in Figure 0
- **Step 1: Generated a new RSA private key for the root certificate.**
  - Command: `openssl genrsa -out rootCA.key 2048`
- **Step 2: Created a new root certificate using the root key.**
  - Command: `openssl req -x509 -new -nodes -key rootCA.key -days 1024 -out rootCA.pem -config ca.cnf`
  - Details: Used the configuration file ca.cnf for certificate details.
    - Ca.cnf shown in Figure 1
- **Step 3: Verified the creation and content of the root certificate.**
  - Command: `openssl x509 -in rootCA.pem -text -noout`
  - Creating a Child Certificate and Signing it with the Root Certificate
- **Step 4: Generated a new RSA private key for the child certificate.**
  - Command: `openssl genrsa -out child.key 2048`
- **Step 5: Created a certificate signing request (CSR) for the child certificate.**
  - Command: `openssl req -new -key child.key -out child.csr -config child.cnf`
    - Child.cnf shown in Figure 2
  - Details: Used the configuration file child.cnf for certificate details.
- **Step 6: Signed the child certificate using the root certificate and key.**
  - Command: `openssl x509 -req -in child.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out child.crt -days 365 -sha256 -extfile child.cnf`
- **Step 7: Verified the creation and content of the child certificate.**
  - Command: `openssl x509 -in child.crt -text -noout`
    - Verification shown in Figure 3

 ca.cnf	11/15/2023 9:41 AM	CNF File	1 KB
 child.cnf	11/15/2023 9:50 AM	CNF File	1 KB
 child.cnf.bak	11/15/2023 9:41 AM	BAK File	1 KB
 child	11/15/2023 9:50 AM	Security Certificate	2 KB
 child.csr	11/15/2023 9:50 AM	CSR File	2 KB
 child.key	11/15/2023 9:50 AM	KEY File	2 KB
 rootCA.key	11/15/2023 9:50 AM	KEY File	2 KB
 rootCA.pem	11/15/2023 9:50 AM	PEM File	2 KB
 rootCA.srl	11/15/2023 9:50 AM	SRL File	1 KB

**Figure 0: File Directory**

```
[ req ]
default_bits = 2048

prompt = no
distinguished_name=req_distinguished_name
req_extensions = v3_req

[ req_distinguished_name ]
countryName=UA
stateOrProvinceName=root region
localityName=root city
organizationName=Market(localhost)
organizationalUnitName=roote department
commonName=market.localhost
emailAddress=root_email@root.localhost

[ alternate_names ]
DNS.1      = market.localhost
DNS.2      = www.market.localhost
DNS.3      = mail.market.localhost
DNS.4      = ftp.market.localhost
DNS.5      = *.market.localhost

[ v3_req ]
keyUsage=digitalSignature
basicConstraints=CA:true
subjectKeyIdentifier = hash
subjectAltName = @alternate_names
```

**Figure 1: Illustrates the configuration settings used for generating a root certificate using OpenSSL. These settings are specified in a configuration file, typically named openssl.cnf, and are categorized into various sections for clarity and organization.**

**The descriptions for each setting is listed below:**

1. [req]: This section defines the default settings for certificate requests.
  - a. default\_bits = 2048: Specifies that the cryptographic strength of the generated key will be 2048 bits, offering a strong level of security.
  - b. prompt = no: Indicates that the user will not be interactively prompted for values; instead, the predefined values in the configuration file will be used.
  - c. distinguished\_name = req\_distinguished\_name: Points to the [req\_distinguished\_name] section, which contains the specifics of the entity for whom the certificate is being requested.
  - d. req\_extensions = v3\_req: Refers to the [v3\_req] section, which includes additional settings for the certificate.
2. [req\_distinguished\_name]: This section outlines the details of the certificate's subject (the entity being certified).
  - a. countryName = UA: Sets the country to Ukraine.

- b. stateOrProvinceName = root region, localityName = root city: Specifies the state/province and locality (city/town) for the certificate.
  - c. organizationName = Market(localhost): The name of the organization requesting the certificate.
  - d. organizationalUnitName = root department: The specific department or division within the organization.
  - e. commonName = market.localhost: The primary domain name that the certificate will be used to secure.
  - f. emailAddress = root\_email@root.localhost: Contact email address for the certificate.
3. [alternate\_names]: Lists alternative domain names that the certificate will secure.
- a. DNS.1 to DNS.5: Includes various subdomains and a wildcard entry (\*.market.localhost) to secure multiple subdomains of market.localhost.
4. [v3\_req]: Specifies extensions for version 3 of X.509 certificates.
- a. keyUsage = digitalSignature: Limits the use of the certificate to digital signatures, which is essential for SSL/TLS.
  - b. basicConstraints = CA:true: Indicates that this certificate can be used to sign other certificates, classifying it as a Certificate Authority (CA).
  - c. subjectKeyIdentifier = hash: A unique identifier for the certificate's key, derived from a hash.
  - d. subjectAltName = @alternate\_names: Refers to the [alternate\_names] section, ensuring that these alternative domain names are included in the certificate.

```
[ req ]
default_bits = 2048

prompt = no
distinguished_name=req_distinguished_name
req_extensions = v3_req

[ req_distinguished_name ]
countryName=US
stateOrProvinceName=Illinois
localityName=Peoria
organizationName=Market(localhost)
organizationalUnitName=roote department
commonName=JacksonLowder
emailAddress=jlowder@mail.bradley.edu

[ alternate_names ]
DNS.1      = market.localhost
DNS.2      = www.market.localhost
DNS.3      = mail.market.localhost
DNS.4      = ftp.market.localhost
DNS.5      = *.market.localhost

[ v3_req ]
keyUsage=digitalSignature
basicConstraints=CA:true
subjectKeyIdentifier = hash
subjectAltName = @alternate_names
```

**Figure 2: displays the configuration settings used for generating a child certificate, based on the child.cnf file. This configuration references and extends the root configuration defined in ca.cnf.**

**Descriptions for each of the configured settings are below:**

1. [req]: Outlines basic requirements for the certificate.
  - a. default\_bits = 2048: Ensures strong security with a 2048-bit key.
  - b. prompt = no: Avoids interactive prompts, using predefined values.
  - c. distinguished\_name = req\_distinguished\_name: Points to the [req\_distinguished\_name] section for certificate subject details.
  - d. req\_extensions = v3\_req: Refers to [v3\_req] for additional certificate settings.
2. [req\_distinguished\_name]: Details the certificate's subject.
  - a. countryName = US, stateOrProvinceName = Illinois, localityName = Peoria: Geographic identifiers for the certificate.
  - b. organizationName = Market(localhost): Organization name.
  - c. organizationalUnitName = roote department: Department within the organization.
  - d. commonName = JacksonLowder: Unique identifier for the certificate, in this case, an individual's name.
  - e. emailAddress = jlowder@mail.bradley.edu: Contact email.

3. [alternate\_names]: Specifies alternative domain names covered by the certificate, similar to the root certificate configuration.
  
4. [v3\_req]: Additional certificate settings.
  - a. keyUsage = digitalSignature: Restricts the certificate's use to digital signatures.
  - b. basicConstraints = CA:true: Signifies that this certificate can sign other certificates, though typically, child certificates would not have this set to true.
  - c. subjectKeyIdentifier = hash, subjectAltName = @alternate\_names: Unique identifiers for the certificate and its alternative names.

```

C:\Users\jacks\Desktop\OpenSSL_Assignment>openssl x509 -req -in child.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out child.crt -days 365 -sha256 -extfile child.cnf
Certificate request self-signature ok
subject=C = US, ST = Illinois, L = Peoria, O = Market(localhost), OU = roote department, CN = JacksonLowder, emailAddress = jlowder@mail.bradley.edu

C:\Users\jacks\Desktop\OpenSSL_Assignment>openssl x509 -in child.crt -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            46:83:02:eb:bd:72:46:a4:7e:7b:e7:1d:ae:5e:86:d8:2c:40:f1:50
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = UA, ST = root region, L = root city, O = Market(localhost), OU = roote department, CN = market.local
host, emailAddress = root_email@root.localhost
        Validity
            Not Before: Nov 15 15:50:41 2023 GMT
            Not After : Nov 14 15:50:41 2024 GMT
        Subject: C = US, ST = Illinois, L = Peoria, O = Market(localhost), OU = roote department, CN = JacksonLowder, em
ailAddress = jlowder@mail.bradley.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (2048 bit)
            Modulus:
                00:e3:a6:6b:8f:08:2a:f1:8e:9c:69:ec:0f:46:b5:
                a9:2e:82:cf:33:64:9e:3d:06:c0:f5:c0:05:7c:7c:
                8d:1b:d0:78:50:2e:9c:16:b3:2b:90:fb:b7:58:fc:
                51:6f:22:e8:e3:8d:61:19:15:42:fe:ef:98:d7:0f:
                d4:80:af:b0:07:88:81:3a:9f:85:8c:cc:a9:72:e8:
                a4:75:3e:ec:63:a9:e4:d1:7d:bd:53:ee:11:63:a3:
                f5:19:37:3e:4a:d5:b1:20:6d:09:4b:8e:9a:20:d6:
                f6:f1:c9:d5:5c:63:07:ef:7f:5d:63:02:0a:f4:d2:
                cd:73:dc:52:b5:c3:2b:75:c1:33:1e:24:54:3f:e2:
                88:4e:a7:c6:06:9c:64:18:61:18:ee:53:30:01:94:
                90:69:07:04:bc:bb:a6:36:87:2d:87:bf:99:d6:6c:
                de:83:e5:51:dd:05:42:b1:d0:0a:4b:df:86:2f:68:
                03:7d:16:9f:1e:cb:3c:0e:ab:35:b1:64:b9:b8:c8:
                65:0a:2f:b5:72:39:32:dc:df:72:26:5b:d0:d0:62:
                ac:25:49:66:c0:4f:de:b4:ca:38:87:e8:26:00:d0:
                6c:ba:b7:d7:9a:0e:1c:f7:bf:4b:be:6f:97:22:45:
                7d:d2:e1:a5:83:ca:07:07:f3:48:21:39:52:29:90:
                35:fb
            Exponent: 65537 (0x10001)
        Signature Algorithm: sha256WithRSAEncryption
        Signature Value:
            4c:07:ba:d6:2f:12:29:d9:88:62:db:56:df:13:22:44:90:6a:
            cc:40:13:af:2d:75:6f:43:44:94:2a:f0:b0:e6:73:ae:c0:ee:
            28:36:e7:3d:c0:e2:81:74:d3:14:81:dd:4f:4a:73:a9:f3:88:
            65:a6:19:ce:44:cb:c0:50:91:c0:a0:b1:b7:74:a7:60:bc:56:
            8b:ea:be:8d:1c:1c:21:f0:e2:58:99:57:6b:59:44:d8:87:cf:
            77:aa:ea:bd:07:f1:aa:e4:f4:e9:a9:8f:32:59:68:cd:ab:e9:
            4e:4b:b4:8e:6f:ac:9a:bd:e2:c7:c7:8f:54:d3:9e:38:92:a8:
            0f:f3:c9:de:f2:81:4e:a2:67:e4:2d:14:05:9e:a2:81:89:71:
            d0:a7:1e:39:e8:70:dc:00:bb:44:b2:19:d3:45:d5:e4:59:0d:
            3e:d9:02:c8:46:59:a4:1b:b2:bb:d2:07:88:7b:da:f2:f4:9a:
            e9:ea:b9:4e:9c:b9:16:17:2c:77:64:b8:82:a4:d9:8d:f5:2f:
            67:f9:2b:2d:08:49:cb:1a:0e:7a:5d:f6:69:ff:4b:ff:b0:0e:
            4e:41:dc:68:cd:df:13:4c:4b:1b:ea:5f:35:e4:cf:0d:28:70:
            ff:2b:1d:b2:4d:78:8d:97:ab:6f:7e:6f:91:da:70:0f:6c:6f:
            31:59:84:c1
    
```

Figure 3: x509, Presents the verification process and the resulting content of the child certificate. This step is crucial to confirm the successful creation and accuracy of the certificate details.

1. Verification Process: Utilizing the OpenSSL command `openssl x509 -in child.crt -text -noout`, the report shows the execution of this command to display the contents of the child certificate in a human-readable format. This step is essential to ensure that all the information in the certificate is correct and aligns with the intended configuration.
2. Content Display: The output displayed in Figure 3 includes various sections of the certificate, such as the subject's details (reflecting information from the [req\_distinguished\_name] section of the child.cnf file), the issuer's details (linking back to the root certificate), and the validity period. It also shows the key usage, basic constraints, and any alternative names specified under [v3\_req] and [alternate\_names]. The common name, JacksonLowder, and the specified email address, jlowder@mail.bradley.edu, are particularly highlighted to demonstrate the personalization of the certificate.



## Extra Credit Section: Explanation of Certificate Extensions

These extensions enhance the functionality and specify the usage parameters of the certificate, as detailed below:

### 1. Subject Alternative Name (SAN):

- a. Configuration:
  - i. [v3\_req]
    - 1. subjectAltName = @alt\_names
  - ii. [alt\_names]
    - 1. DNS.1 = example.com
    - 2. DNS.2 = www.example.com
- b. Purpose: This extension allows the certificate to cover multiple domain names or IP addresses. It is especially important for web servers that need to operate across various domain names or subdomains.

### 2. Key Usage:

- a. Configuration:
  - i. [v3\_req]
    - 1. keyUsage = digitalSignature, keyEncipherment
- b. Purpose: Defines the intended use of the public key within the certificate, such as for digital signatures or key encipherment. This specification is crucial for determining the scope of the certificate's application.

### 3. Extended Key Usage:

- a. Configuration:
  - i. [v3\_req]
    - 1. extendedKeyUsage = serverAuth, clientAuth
- b. Purpose: Specifies the specific contexts in which the certificate's public key can be utilized. This includes applications like server authentication and client authentication, providing clarity on the operational environment of the certificate.

### 4. Basic Constraints:

- a. Configuration:
  - i. [v3\_req]
    - 1. basicConstraints = CA:FALSE
- b. Purpose: Indicates whether the certificate is a Certificate Authority (CA) certificate. This is a critical distinction for intermediate certificates, delineating their role in the certificate hierarchy.

## 5. Certificate Policies:

- a. Configuration:
  - i. [v3\_req]
    - 1. certificatePolicies = @cert\_policies
  - ii. [cert\_policies]
    - 1. policyIdentifier = 1.2.3.4.5.6.7
    - 2. CPS.1 = <http://www.example.com/cps>
- b. Purpose: Includes a set of policy information terms, outlining the policies under which the certificate was issued. This extension provides an additional layer of information about the certificate's issuance criteria and standards.

Each of these extensions plays a vital role in defining the operational parameters and security features of the certificate, ensuring its suitability and compliance for its intended applications.

## Conclusion

In conclusion, this report successfully navigates the intricacies of digital certificate creation and management using OpenSSL. The process began with the generation of a root self-signed certificate, establishing a foundational trust anchor. Key elements of the configuration file, `ca.cnf`, were meticulously outlined, underscoring how each parameter contributes to the security and identity of the certificate.

The creation of a child certificate followed, where personalization played a critical role. By embedding specific user details and domain names into the certificate, we demonstrated how a child certificate inherits trust from its root while catering to individual requirements. The verification of the child certificate was a critical step, ensuring the accuracy and integrity of the certificate's contents and its alignment with the intended configurations.

The exploration of advanced certificate extensions in the extra credit section added depth to the project. It highlighted the versatility and robustness of OpenSSL in handling various certificate features, such as Subject Alternative Names, Key Usage, Extended Key Usage, Basic Constraints, and Certificate Policies. These extensions showcased the potential for tailoring certificates to specific security needs and contexts.

Overall, this project exemplified the complex yet essential nature of digital certificates in establishing secure communications. It provided a practical demonstration of the skills and knowledge required to manage digital certificates effectively, reflecting a comprehensive understanding of the key principles and practices in digital security.