TRABAJO FIN DE GRADO

UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

PLANTILLAS DE CONFIGURACIÓN OPENSSL

Autor:

D. Marc Hernández Montesinos

Directora:

Dra. Dña. *Angélica Guzmán Ponce*

Murcia, Junio de 2026

TRABAJO FIN DE GRADO

UCAM
UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

PLANTILLAS DE CONFIGURACIÓN OPENSSL

Autor:

D. Marc Hernández Montesinos

Directora:

Dra. Dña. *Angélica Guzmán Ponce*

Murcia, Junio de 2026

https://tfg.marchernandez.es/videos/video-demostrativo.mp4

ÍNDICE

PLANTILLAS DE CONFIGURACIÓN OPENSSL

Perfiles X.509 v3 con KU, EKU, AIA, CDP y Policy

Documento separado del Trabajo de Fin de Grado *"Diseño e implementación de un sistema integrado de PKI y SSO para organizaciones pequeñas"* de Marc Hernández Montesinos (UCAM, Grado en Ingeniería Informática).

Campo	Valor
Documento	Plantillas de Configuración OpenSSL
Versión	1.0
Fecha	1 junio 2026
URL pública	https://tfg.marchernandez.es/manuales/Manual_Plantillas_OpenSSL.pdf
TFG asociado	https://tfg.marchernandez.es

Este manual conserva la numeración interna original (sección F.x) por trazabilidad con las versiones previas del documento. Las referencias cruzadas que apuntan al cuerpo del TFG (capítulos 1-10, anexos A-D) se mantienen tal cual y son válidas frente al PDF principal del TFG.

Este anexo recoge, comentadas, las plantillas de configuración de OpenSSL que el sistema utiliza para emitir cada tipo de certificado X.509 v3 descrito en el capítulo 7. Las plantillas son el artefacto técnico más importante de la PKI: en ellas se declaran de forma estructural -y por tanto, verificable por cualquier validador X.509 conforme- las extensiones que determinan los usos permitidos del certificado, los puntos de validación de revocación y la jerarquía de confianza. Cualquier auditoría seria del sistema debe poder partir de estos ficheros sin necesidad de leer ni una línea del código PHP.

El anexo se estructura en tres bloques. Sección F.1 presenta una visión global de las plantillas y de su relación con la jerarquía PKI. Sección F.2 a Sección F.9 recogen, en orden de uso, la plantilla raíz, la plantilla parametrizada de intermedia y los seis perfiles X.509 de usuario final. Sección F.10 y Sección F.11

muestran ejemplos reales de plantillas dinámicas generadas por `PKIEngine` durante operación. Sección F.12 cierra el anexo con una tabla comparativa que resume *Key Usage*, *Extended Key Usage*, validez y restricciones de cada perfil. Todas las plantillas se conservan bajo `pki/public/config/openssl/` en el repositorio del proyecto y están disponibles para inspección directa.

F.1 Visión general

Las plantillas se organizan en tres niveles, que coinciden con la jerarquía PKI descrita en Sección 7.4.1:

Nivel	Plantilla	Quién la usa	Resultado
Raíz	`root_openssl.cnf`	Operador en la generación inicial de la CA raíz	Certificado autofirmado `root_ca.crt`
Intermedia	`intermediate_openssl.tpl.cnf` (con `${CA_SLUG}`)	`PKIEngine::createIntermediateCA` al crear cada CA intermedia	CSR de la intermedia y, tras firma de la raíz, el certificado `intermediate_ca.crt`
Usuario final	`profiles/*.cnf` (seis perfiles)	`PKIEngine::signCertificate` al emitir cada certificado de usuario	Certificado de usuario firmado por la intermedia correspondiente

Tabla F.1. Niveles de plantilla, consumidor y artefacto resultante.

Las tres reglas estructurales transversales que comparten todas las plantillas son:

1. **`default_md = sha256`**: SHA-256 como algoritmo de resumen por defecto en todas las firmas. La excepción es la plantilla operativa de la raíz instalada en producción (`openssl_int_sign_*.cnf`, Sección F.10), que usa `sha384` cuando firma intermedias para reforzar el eslabón más sensible de la cadena.

2. `copy_extensions = copy` en las intermedias (y `none` en la raíz): las intermedias copian las extensiones del CSR del solicitante (necesario para que los SAN del usuario final se preserven); la raíz, por el contrario, descarta todas las extensiones del CSR de la intermedia para evitar inyecciones.

3. Tres URLs canónicas presentes en todo certificado emitido: `AIA` (descarga del certificado emisor), `OCSP` (responder en `ocsp.marchernandez.es`) y `crlDistributionPoints` (CRL en `crl.marchernandez.es`). Esto permite que cualquier validador externo descubra automáticamente los puntos de validación sin configuración previa.

Las variables parametrizables en las plantillas son:

- `${CA_SLUG}`: identificador corto de la CA intermedia (p. ej. `intermediate-ev`, `racv-ca`, `codesigning-ca`); se sustituye en el momento de generar el `.cnf` final concreto para cada intermedia.

- `${CA_ID}`: identificador numérico de la CA intermedia en la base de datos (no usado en el `.cnf` estático pero sí en los `.cnf` dinámicos de Sección F.10-F.11).

F.2 Plantilla raíz: `root_openssl.cnf`

La plantilla raíz se ejecuta una sola vez en el momento de inicializar la jerarquía PKI. Crea la CA raíz autofirmada, define las políticas estrictas que aplicará al firmar a sus intermedias y declara las extensiones críticas que esas intermedias llevarán grabadas.

```
#
####################################################################
# OpenSSL Configuration - Root CA
# Omnipresence TrustCA EV Root CA
#
####################################################################
# OFFLINE CA - No debe estar en el servidor de producción
#
####################################################################

[ ca ]
default_ca = CA_default

[ CA_default ]
dir               = /var/pki/root
database          = $dir/db/index.txt
serial            = $dir/db/serial
crlnumber         = $dir/db/crlnumber
new_certs_dir     = $dir/newcerts
private_key       = $dir/private/root.key
certificate       = $dir/certs/root.crt
default_md        = sha256
policy            = policy_strict
email_in_dn       = no
nameopt           = default_ca
certopt           = default_ca
default_days      = 7300
crl_days          = 180
unique_subject    = no
copy_extensions   = none
preserve          = no
x509_extensions   = v3_intermediate_ca
```

Puntos destacados de la sección `[ CA_default ]`:

- `default_days = 7300` → 20 años de validez para los certificados que la raíz firme (sus intermedias). En la práctica operativa actual el valor efectivo aplicado por `PKIEngine` es 3.650 días (10 años), definido en el `.cnf` dinámico de Sección F.10; el 7.300 de la plantilla queda como referencia para una eventual emisión con vigencia máxima.
- `crl_days = 180` → frecuencia de regeneración de la CRL de la raíz; baja porque las revocaciones a nivel raíz son extremadamente raras.
- `copy_extensions = none` → la raíz no copia las extensiones del CSR de la intermedia, sino que aplica las suyas definidas en `[ v3_intermediate_ca ]`. Esto evita que un atacante con acceso al CSR pueda intentar inyectar extensiones no autorizadas.

- `policy = policy_strict` → al firmar a una intermedia, la raíz exige coincidencia exacta de `countryName` y `organizationName` con los suyos propios.

Las dos secciones de extensiones críticas que la raíz puede aplicar:

```
[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, keyCertSign, cRLSign

[ v3_intermediate_ca ]
# pathlen:0 -> las intermedias firman únicamente certificados de usuario final,
# nunca nuevas sub-CAs (defensa estructural conforme a RFC 5280 Sección 6.1.4 (k)).
# La jerarquía operativa es: raíz -> intermedia -> certificado final.
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, keyCertSign, cRLSign
crlDistributionPoints = URI:http://crl.marchernandez.es/root/ca.crl
```

`[ v3_ca ]` es el bloque que la propia raíz aplica sobre sí misma durante la auto-firma inicial: `CA:true` sin `pathlen` -porque la raíz no tiene restricción de profundidad- y `keyCertSign, cRLSign` críticos para que cualquier validador conforme a RFC 5280 Sección 4.2.1.3 rechace cualquier intento de usar la raíz para fines distintos.

`[ v3_intermediate_ca ]` es el bloque que la raíz aplica sobre las intermedias que firma. La línea clave aquí es `basicConstraints = critical, CA:true, pathlen:0`: cualquier validador conforme a RFC 5280 Sección 6.1.4 (k) rechazará una nueva CA firmada por una intermedia, porque su propio certificado lo prohíbe. La jerarquía operativa queda así cerrada en dos niveles -raíz → intermedia → certificado de usuario- sin posibilidad estructural de sub-intermedias, ni siquiera en el escenario peor (intermedia comprometida). La defensa es deliberadamente más estricta que la del enfoque convencional con `pathlen:1`: a cambio de perder la flexibilidad de jerarquías de tres niveles -que el sistema no necesita- se cierra una vía de escalada de privilegios que sí podría aprovechar un atacante.

La sección OCSP signer declara el bloque que se usa al emitir el certificado dedicado del *responder* OCSP:

```
[ ocsp ]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

Los dos puntos clave son `CA:FALSE` (el *signer* OCSP no puede firmar nuevos certificados, sólo respuestas OCSP) y `extendedKeyUsage = critical, OCSPSigning`. La criticidad de `extendedKeyUsage` es deliberada: un cliente que reciba esta clave debe rechazar usarla para cualquier otro propósito.

F.3 Plantilla parametrizada de intermedia: `intermediate_openssl.tpl.cnf`

La plantilla de intermedia es la pieza más extensa de la configuración. Se utiliza como plantilla que `PKIEngine` materializa en un `.cnf` concreto por cada CA intermedia, sustituyendo `${CA_SLUG}` por el identificador real (p. ej. `intermediate-ev`).

```
#
# ====================================================================
# OpenSSL Configuration - Intermediate CA Template
# Plantilla para CAs Intermedias
#
# ====================================================================
# Variables a reemplazar:
# ${CA_SLUG} - slug de la CA (ej: intermediate-ev, racv-ca, codesigning-ca)
#
# ====================================================================

[ ca ]
default_ca = CA_default

[ CA_default ]
dir               = /var/pki/intermediates/${CA_SLUG}
database          = $dir/db/index.txt
serial            = $dir/db/serial
crlnumber         = $dir/db/crlnumber
new_certs_dir     = $dir/newcerts
private_key       = $dir/private/ca.key
certificate       = $dir/certs/ca.crt
default_md        = sha256
policy            = policy_loose
email_in_dn       = no
nameopt           = default_ca
certopt           = default_ca
default_days      = 825
crl_days          = 7
unique_subject    = no
copy_extensions   = copy
preserve          = no
```

Diferencias clave respecto a la plantilla raíz:

- `default_days = 825` → 825 días (aproximadamente 27 meses), un valor seguro para la mayoría de certificados de usuario final. El perfil `serverAuth` lo recortará a 398 días por exigencia del CA/Browser Forum.
- `crl_days = 7` → CRL semanal a nivel intermedia (la rotación real en producción es cada 4 h, vía `systemd timer`; ese `7` queda como respaldo).
- `policy = policy_loose` → al firmar a usuarios finales, las intermedias no exigen coincidencia de `countryName` u `organizationName`, lo que permite emitir certificados a cualquier solicitante autorizado.
- `copy_extensions = copy` → las intermedias sí copian las extensiones del CSR del usuario. Esto es necesario para preservar los SAN (DNS y correos) que el usuario declara. La validación de qué SAN son aceptables

se realiza en `PKIEngine::signCertificate` antes de invocar OpenSSL, no en la propia plantilla.

La extensión más crítica de toda la PKI vive en `[ v3_intermediate_ca ]`:

```
[ v3_intermediate_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, keyCertSign, cRLSign
authorityInfoAccess =
caIssuers;URI:http://ca.marchernandez.es/intermediates/${CA_SLUG}.crt
crlDistributionPoints = URI:http://crl.marchernandez.es/${CA_SLUG}/ca.crl
```

`pathlen:0` es la línea más importante del sistema PKI completo. Cualquier validador conforme a RFC 5280 Sección 6.1.4 (k) que valide una cadena en la que aparece una intermedia con `pathlen:0` rechazará cualquier certificado de CA firmado por ella, independientemente de la corrección de la firma o de la fecha. Es una defensa estructural: aun en el escenario peor (intermedia totalmente comprometida con la *passphrase* de su clave extraída), el atacante no puede convertir esa intermedia en raíz de una jerarquía paralela.

La plantilla también define tres perfiles de extensiones de usuario final por defecto (`[ usr_cert ]`, `[ server_cert ]`, `[ server_cert_must_staple ]`), que sirven como *fallback* si por algún motivo el flujo de emisión no carga uno de los `.cnf` de perfil específicos (Sección F.4-F.9). En operación normal, los perfiles específicos siempre prevalecen.

F.4 Perfil `clientAuth`

Este perfil emite certificados para autenticación de cliente (TLS mutuo, login con certificado en aplicaciones web, S/MIME parcial). Es el perfil más usado en el sistema, dado que es el que reciben los usuarios cuando solicitan un certificado para identificarse en otros servicios.

```
[ usr_cert ]
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "Omnipresence TrustCA Client Authentication Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature, keyAgreement
extendedKeyUsage = clientAuth, emailProtection
authorityInfoAccess =
caIssuers;URI:http://ca.marchernandez.es/intermediates/${CA_SLUG}.crt,
OCSP;URI:http://ocsp.marchernandez.es/${CA_SLUG}
crlDistributionPoints = URI:http://crl.marchernandez.es/${CA_SLUG}/ca.crl
```

Los puntos clave:

- `keyUsage = critical, digitalSignature, keyAgreement` → la clave puede firmar (autenticación) y acordar claves de sesión (necesario para algunos modos TLS), pero no puede cifrar bulk data ni firmar otros certificados.
- `extendedKeyUsage = clientAuth, emailProtection` → uso permitido como cliente TLS y, opcionalmente, como remitente de correo firmado. La validez por defecto, heredada de la intermedia, es 825 días.
- `nsCertType = client, email` → extensión Netscape obsoleta pero todavía verificada por algunos productos heredados. Se mantiene como compatibilidad sin coste.

F.5 Perfil `serverAuth`

Este perfil emite certificados para servidores TLS: HTTPS, IMAP/SMTP TLS, MQTT TLS, etc. La diferencia más relevante respecto a `clientAuth` es la validez, acortada a 398 días, por imposición del CA/Browser Forum desde septiembre de 2020 para todos los certificados TLS de servidor.

```
[ usr_cert ]
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "Omnipresence TrustCA Server Authentication Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth
authorityInfoAccess =
caIssuers;URI:http://ca.marchernandez.es/intermediates/${CA_SLUG}.crt,
OCSP;URI:http://ocsp.marchernandez.es/${CA_SLUG}
crlDistributionPoints = URI:http://crl.marchernandez.es/${CA_SLUG}/ca.crl

# Must-Staple opcional (descomentar si se requiere)
# tlsfeature = status_request
```

Las particularidades:

- `keyUsage = critical, digitalSignature, keyEncipherment` → necesario para el *handshake* TLS RSA legacy (en TLS 1.3 puro sólo se necesita `digitalSignature`, pero `keyEncipherment` se conserva por compatibilidad).
- `extendedKeyUsage = serverAuth` exclusivamente → un certificado emitido bajo este perfil no podrá usarse para autenticación de cliente ni para correo.
- `tlsfeature = status_request` → comentado por defecto; si se activa, el certificado lleva la extensión Must-Staple (RFC 7633) que obliga al servidor a entregar respuesta OCSP grapeada en cada *handshake*. Se ofrece, pero no se fuerza, porque despliegues sin `OCSP stapling` configurado quedarían inutilizables.

F.6 Perfil `codeSigning`

Este perfil emite certificados para firmar software ejecutable (binarios Windows, paquetes macOS, ejecutables Linux con `osslsigncode`). Es uno de los perfiles más restrictivos.

```
[ usr_cert ]
basicConstraints = CA:FALSE
nsCertType = objsign
nsComment = "Omnipresence TrustCA Code Signing Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = codeSigning
authorityInfoAccess =
caIssuers;URI:http://ca.marchernandez.es/intermediates/${CA_SLUG}.crt,
OCSP;URI:http://ocsp.marchernandez.es/${CA_SLUG}
crlDistributionPoints = URI:http://crl.marchernandez.es/${CA_SLUG}/ca.crl
```

Los aspectos restrictivos del perfil son operacionales, no del `.cnf` propiamente dicho: `PKIEngine::signCertificate` valida que la solicitud cumple **dos requisitos** adicionales antes de firmar:

1. Longitud mínima de clave RSA ≥ 4.096 bits, o curva ECDSA P-256/P-384 (alineado con la *baseline* de Microsoft Authenticode 2022).
2. Aprobación obligatoria por un administrador: el flujo de emisión no permite auto-aprobación para este perfil.

El `keyUsage` está restringido únicamente a `digitalSignature` -no se incluye `keyEncipherment`- porque firmar código sólo necesita firmar; cualquier capacidad adicional sería un riesgo innecesario.

F.7 Perfil `documentSigning`

Este perfil emite certificados para firma electrónica de documentos (PDF, XML, DOCX), una funcionalidad cada vez más demandada por organizaciones que necesitan firmar facturas electrónicas, contratos o actas.

```
[ usr_cert ]
basicConstraints = CA:FALSE
nsComment = "Omnipresence TrustCA Document Signing Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature, nonRepudiation
extendedKeyUsage = emailProtection
authorityInfoAccess =
caIssuers;URI:http://ca.marchernandez.es/intermediates/${CA_SLUG}.crt,
OCSP;URI:http://ocsp.marchernandez.es/${CA_SLUG}
crlDistributionPoints = URI:http://crl.marchernandez.es/${CA_SLUG}/ca.crl

# OID propietario para Document Signing (opcional - descomentar y ajustar)
# extendedKeyUsage = emailProtection, 1.3.6.1.4.1.99999.1
```

La extensión clave es `keyUsage = critical, digitalSignature, **nonRepudiation**`. El bit `nonRepudiation` (también llamado `contentCommitment` en RFC 5280) declara que la firma producida por esta clave es vinculante y se puede usar como prueba de que el titular firmó el contenido, no sólo como autenticación efímera. Este matiz tiene relevancia legal: jurisdicciones europeas que reconocen firma electrónica avanzada o cualificada (eIDAS) requieren este bit explícito.

El `extendedKeyUsage = emailProtection` actúa como EKU genérica para uso en aplicaciones cliente; algunas suites de firma de documentos (Adobe Acrobat, LibreOffice) aceptan certificados con `emailProtection` para firma PDF, mientras que otras exigen un OID propietario específico (que se puede activar descomentando la línea final del fichero).

F.8 Perfil `smimeEmail`

Este perfil emite certificados para firma y cifrado de correo electrónico S/MIME según RFC 8551. Es similar al `clientAuth` pero con su EKU restringida exclusivamente a `emailProtection`.

```
[usr_cert]
basicConstraints       = CA:FALSE
nsCertType             = client, email
nsComment              = "Omnipresence TrustCA S/MIME Email Certificate"
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
keyUsage               = critical, digitalSignature, keyEncipherment
extendedKeyUsage       = emailProtection
authorityInfoAccess    =
caIssuers;URI:http://ca.marchernandez.es/intermediates/${CA_SLUG}.crt,
OCSP;URI:http://ocsp.marchernandez.es/${CA_SLUG}
crlDistributionPoints  = URI:http://crl.marchernandez.es/${CA_SLUG}/ca.crl
```

Los puntos diferenciadores son:

- `keyUsage = critical, digitalSignature, keyEncipherment` → necesario para el doble uso de S/MIME (firma del mensaje + cifrado de la clave de sesión que cifra el cuerpo).
- `extendedKeyUsage = emailProtection` único → un certificado bajo este perfil no puede usarse como cliente TLS, ni para firma de código, ni para servidor.

- AIA y CRL Distribution Points incluidos explícitamente en la plantilla. En el flujo normal de emisión, `PKIEngine::signCertificate` reconstruye el `.cnf` final con AIA, OCSP y CDP añadidos por código (líneas 1124-1125 de `pki/lib/PKIEngine.php`), garantizando que el certificado emitido los lleva con independencia del perfil. La inclusión en el `.cnf` estático aporta dos beneficios: que la plantilla sea autosuficiente si alguien la ejecuta manualmente fuera del flujo, y que la documentación sea fiel a lo que se graba en el certificado final.

La inclusión del correo electrónico en `subjectAltName` se realiza en el momento de la solicitud, no en la plantilla; el SAN se extrae del CSR enviado por el usuario y se preserva gracias al `copy_extensions = copy` de la intermedia.

F.9 Perfil `vpn`

Este perfil emite certificados específicamente para autenticación de clientes VPN (OpenVPN, IPSec/IKEv2). La particularidad técnica relevante es la inclusión de un OID específico (`1.3.6.1.5.5.8.2.2`) en `extendedKeyUsage`.

```
[usr_cert]
basicConstraints       = CA:FALSE
nsCertType             = client
nsComment              = "Omnipresence TrustCA VPN Certificate"
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid,issuer
keyUsage               = critical, digitalSignature, keyAgreement
extendedKeyUsage       = clientAuth, 1.3.6.1.5.5.8.2.2
authorityInfoAccess    =
caIssuers;URI:http://ca.marchernandez.es/intermediates/${CA_SLUG}.crt,
OCSP;URI:http://ocsp.marchernandez.es/${CA_SLUG}
crlDistributionPoints  = URI:http://crl.marchernandez.es/${CA_SLUG}/ca.crl
```

El OID `1.3.6.1.5.5.8.2.2` corresponde a **`id-kp-ipsecIKE`** (IKE Intermediate / IPsec Internet Key Exchange), introducido en el RFC 4945 Sección 5.1.3.12. Su presencia permite que implementaciones IPsec estrictas -que rechazan certificados sin EKU IPsec- acepten el certificado. La combinación `clientAuth + id-kp-ipsecIKE` es la postura recomendada para certificados de cliente VPN modernos, que se usan tanto en escenarios de OpenVPN (más laxo con la EKU) como en IPsec/IKEv2 (que la exige explícitamente). Las dos últimas líneas (AIA y CDP) siguen el mismo criterio que en el perfil S/MIME: autosuficiencia del `.cnf` aunque el flujo de emisión normal las componga también por código.

F.10 Plantilla operativa de firma de intermedia

Las plantillas estáticas anteriores se complementan con plantillas dinámicas que `PKIEngine` genera *al vuelo* y deja en `pki/ca/{role}/openssl_int_sign_<id>.cnf` cada vez que ejecuta una operación. Estas plantillas materializan rutas absolutas reales del despliegue de producción y son la evidencia más directa de lo que efectivamente ha ejecutado OpenSSL. El ejemplo siguiente corresponde a la firma de una CA intermedia por parte de la raíz:

```
[ca]
default_ca = CA_default

[CA_default]
dir               =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root
certs             =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root
crl_dir           =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root
database          =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/index.txt
new_certs_dir     =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root
certificate       =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/root_ca.pem
private_key       =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/root_ca.key
serial            =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/serial
crlnumber         =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/crlnumber
crl               =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/crl.pem
RANDFILE          =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/.rand
default_days      = 3650
default_crl_days  = 30
default_md        = sha384
preserve          = no
policy            = policy_loose
unique_subject    = no
```

Diferencias respecto a la plantilla estática `root_openssl.cnf` que merecen comentario:

- Las rutas son absolutas y reales del VPS de producción (`/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/root/`), no las rutas teóricas de la plantilla original (`/var/pki/root`). La plantilla estática es declarativa; esta es ejecutable.

- `default_md = sha384` en lugar de `sha256` para la operación de firma de intermedias. Esta elevación al SHA-384 en el eslabón más sensible de la cadena es una decisión consciente del operador: la raíz sólo se usa para firmar intermedias y para emitir la CRL raíz; ambas operaciones se benefician de un resumen más fuerte sin penalizar el rendimiento (la raíz firma muy pocos objetos al año).
- `default_days = 3650` → las intermedias se emiten con 10 años de validez, alineado con la práctica habitual de operadores PKI maduros. Es menos que los 7.300 días teóricos de la plantilla original, lo que se considera más prudente.

El bloque de extensiones de intermedia incorpora `certificatePolicies` y `authorityInfoAccess` que la plantilla estática no llevaba:

```
[v3_intermediate_ca]
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints       = critical, CA:true, pathlen:1
keyUsage               = critical, digitalSignature, cRLSign, keyCertSign
certificatePolicies    = @pol_section
authorityInfoAccess    =
OCSP;URI:http://ocsp.marchernandez.es,caIssuers;URI:http://ca.marchernandez.e
s/root.crt
crlDistributionPoints  = URI:http://crl.marchernandez.es/root.crl

[pol_section]
policyIdentifier       = 2.5.29.32.0
CPS.1                  = https://pki.marchernandez.es/dpc/2
```

`certificatePolicies` con `policyIdentifier = 2.5.29.32.0` (el OID `anyPolicy`) declara que la intermedia opera bajo cualquier política aplicable. El campo `CPS.1` apunta a la Declaración de Prácticas de Certificación específica de la rama, alojada en `https://pki.marchernandez.es/dpc/2`, donde se documenta el alcance y limitaciones operacionales de la intermedia.

Nota honesta sobre las intermedias actualmente desplegadas. El fichero histórico mostrado arriba refleja una emisión anterior al endurecimiento descrito en este anexo: la intermedia que firmó este `.cnf` quedó con `basicConstraints = critical, CA:true, pathlen:1` grabado en su certificado, lo que estructuralmente permitiría firmar una sub-CA. A partir de la consolidación documental de este TFG, `PKIEngine::createIntermediateCA` ha sido modificado para forzar `pathlen:0`

en cualquier nueva intermedia (líneas 902-905 de `pki/lib/PKIEngine.php`), alineándose con la plantilla estática `root_openssl.cnf` (Sección F.2). El cambio aplica únicamente a las intermedias futuras: las intermedias activas que ya llevan `pathlen:1` grabado conservarán ese valor hasta su revocación y reemisión - procedimiento documentado en el anexo C como rotación programada. La defensa estructural sobre la jerarquía operativa no se ve comprometida mientras tanto, porque el flujo de aplicación (`signCertificate`) nunca emite certificados de CA, y ningún consumidor del sistema obtiene una clave de intermedia.

F.11 Plantilla operativa de generación de CRL

Las CRL se generan cada 4 h vía `pki/cron/generate_crl.php`, que produce un `.cnf` específico por intermedia y ejecuta `openssl ca -gencrl` contra él. Un ejemplo real:

```
[ca]
default_ca = CA_default

[CA_default]
dir               =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/intermediate_3
database          =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/intermediate_3/index
.txt
new_certs_dir     =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/intermediate_3
certificate       =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/intermediate_3/inter
mediate_ca.pem
private_key       =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/intermediate_3/inter
mediate_ca.key
serial            =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/intermediate_3/seria
l
crlnumber         =
/var/www/vhosts/marchernandez.es/pki.marchernandez.es/ca/intermediate_3/crlnu
mber
default_days      = 365
default_crl_days  = 1
default_md        = sha256
preserve          = no
policy            = policy_loose
unique_subject    = no
```

Puntos relevantes para la lectura:

- `default_crl_days = 1` en este `.cnf` operativo: el valor declarado en el fichero indica que el campo `nextUpdate` de la CRL se calcula 24 h después de la fecha actual. Conviene aclarar la cadena de decisiones que opera aquí:
    - `PKIEngine::generateCRL` lee `crl.validity_hours` desde la configuración aplicativa (`pki_config` y `config/app.php`, con valor `4` en producción).
    - A la hora de invocar OpenSSL, calcula `-crldays (int)ceil(validity_hours / 24) ?: 1`, lo que para 4 h se redondea a 1 día.
    - Como consecuencia, el `nextUpdate` del DER firmado es +24 h, no +4 h, *aunque el parámetro aplicativo diga 4 h*. La frecuencia de regeneración la marca el *timer* de systemd (cada 4 h) y la validez del DER la marca este `-crldays`, lo que se traduce en un margen estructural de hasta seis rotaciones perdidas antes de quedar sin CRL válida.
    - Esto se discute en detalle en Sección 7.4.5 y Sección 7.8 del cuerpo, y se documenta como mejora futura (Sección 9.4) la posibilidad de pasar a `-startdate`/`-enddate` para reducir la validez del DER a 4 h reales.
- El comando efectivo que se ejecuta es `openssl ca -config <fichero> -gencrl -out <ruta>.crl`, e inmediatamente después la CRL se replica al *vhost* público de `crl.marchernandez.es` en formatos DER y PEM.

F.12 Tabla comparativa de los seis perfiles

La tabla F.2 resume las extensiones críticas y la validez por defecto de los seis perfiles X.509 v3 de usuario final. La validez efectiva final aplicada por `PKIEngine` es el menor valor entre el solicitado, el aprobado por el administrador y el `validity_days` declarado en la base de datos para esa plantilla.

Perfil	`keyUsage`	`extendedKeyUsage`	Validez por defecto	Restricciones específicas
clientAuth	digitalSignature, keyAgreement	clientAuth, emailProtection	825 días	Email obligatorio en SAN
serverAuth	digitalSignature, keyEncipherment	serverAuth	**398 días** (límite CA/Browser Forum)	DNS o IP obligatorio en SAN
codeSigning	digitalSignature	codeSigning	730 días	RSA ≥ 4 096 bits o ECDSA P-256/P-384; aprobación admin obligatoria
documentSigning	digitalSignature, **nonRepudiation**	emailProtection (+ OID propietario opcional)	1 095 días	Aprobación admin obligatoria; uso vinculante (eIDAS)
smimeEmail	digitalSignature, keyEncipherment	emailProtection	825 días	Email obligatorio en SAN
vpn	digitalSignature, keyAgreement	clientAuth, **id-kp-ipsecIKE** (1.3.6.1.5.5.8.2.2)	730 días	Email opcional; CN identifica el cliente VPN

Tabla F.2. Extensiones críticas, validez y restricciones por perfil X.509 v3.

Las tres extensiones presentes en todos los perfiles y omitidas de la tabla por brevedad son:

- `subjectKeyIdentifier = hash` y `authorityKeyIdentifier = keyid,issuer`, que permiten la construcción rápida de la cadena por parte de los validadores.
- `basicConstraints = CA:FALSE`, que prohíbe que el certificado actúe como CA bajo ninguna circunstancia.
- Las tres URLs canónicas (`AIA caIssuers`, `AIA OCSP`, `crlDistributionPoints`), añadidas por composición desde la plantilla de intermedia.

F.13 Verificación operativa de un certificado emitido

Como complemento operativo, el comando estándar para verificar manualmente que un certificado emitido por el sistema lleva las extensiones correctas es:

```
openssl x509 -in usuario.crt -noout -text | grep -A4 -E "X509v3 (Key
Usage|Extended Key Usage|Basic Constraints|CRL Distribution|Authority
Information Access|Certificate Policies)"
```

Una salida correcta para un certificado `clientAuth` recién emitido debe mostrar **todos** los bloques siguientes, en este orden o equivalente:

```
X509v3 Basic Constraints: critical
    CA:FALSE
X509v3 Key Usage: critical
    Digital Signature, Key Agreement
X509v3 Extended Key Usage:
    TLS Web Client Authentication, E-mail Protection
X509v3 CRL Distribution Points:
    Full Name:
      URI:http://crl.marchernandez.es/<ca_slug>/ca.crl
X509v3 Certificate Policies:
    Policy: 1.3.6.1.4.1.55032.1.1
      CPS: https://pki.marchernandez.es/dpc/<ca_id>
Authority Information Access:
    OCSP - URI:http://ocsp.marchernandez.es/<ca_slug>
    CA Issuers - URI:http://ca.marchernandez.es/intermediates/<ca_slug>.crt
```

Cualquier desviación indica que la cadena ha sido firmada por una jerarquía distinta o que las plantillas del despliegue han sido modificadas.

Procedimiento de verificación recomendado tras emisión. Para los perfiles `smimeEmail` y `vpn`, recientemente endurecidos con AIA y CDP explícitos en su `.cnf` (Sección F.8 y Sección F.9), conviene emitir un certificado de prueba por perfil en el entorno de *staging* y ejecutar el comando anterior. La presencia de los bloques `Authority Information Access` y `CRL Distribution Points` debe confirmarse antes de promover el perfil a producción, especialmente para los perfiles que históricamente no los llevaban en su `.cnf` estático y dependían de la composición dinámica de `PKIEngine`. El resultado de esta verificación se incorpora al anexo H (vulnerabilidades evaluadas y mitigadas) como evidencia operativa del control "AIA/CDP presentes en todo certificado emitido".

Las plantillas completas, incluyendo cualquier evolución posterior al cierre del documento, se mantienen actualizadas en `pki/public/config/openssl/` del repositorio del proyecto. El anexo G (código relevante seleccionado) muestra las funciones de `PKIEngine` que invocan estas plantillas en cada operación.