
The openssl Method: Complete Manual Control

These steps will guide you through creating your own ECDSA CA and signing an ECDSA server certificate.

Part 1: Create Your ECDSA Certificate Authority (CA)

1. Generate the CA's private key:
   This creates a high-quality ECDSA private key.
   Bash
   openssl ecparam -name prime256v1 -genkey -out ca.key

2. Create the self-signed CA root certificate:
   This uses the key to create the ca.crt file that clients will trust.
   Bash
   openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out ca.crt -subj "/CN=MinIO Lab ECDSA CA"

Part 2: Create Your ECDSA Server Certificate

1. Generate the server's private key:
   This creates the private.key that MinIO will use.
   Bash
   openssl ecparam -name prime256v1 -genkey -out private.key

2. Create a configuration file for domain names:
   openssl requires a file to list the domain names for the certificate. This command creates it for you.
   Bash
   cat > san.conf <<EOF
   [req]
   distinguished_name = req_distinguished_name
   req_extensions = v3_req
   [req_distinguished_name]
   [v3_req]
   subjectAltName = @alt_names
   [alt_names]
   DNS.1 = *.crjv.local

DNS.2 = crjv.local
DNS.3 = localhost
IP.1 = 127.0.0.1
EOF

3. Create a Certificate Signing Request (CSR):
   This bundles the server's public key and domain info into a formal request.
   Bash
   openssl req -new -key private.key -out server.csr -subj "/CN=minio.crjv.local" -config san.conf

4. Sign the server certificate with your CA:
   This is the final step, where your CA signs the request, creating the public.crt.
   Bash
   openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out public.crt -days 730 -sha256 -extfile san.conf -extensions v3_req

---

Part 3: Final Files and Validation ✅

You have now created all the necessary files. See the bottom where we need to convert private.key to PKCS#8.

- **Your final files are:** ca.crt, public.crt, and private.key.
- You can safely delete the intermediate files: ca.key, server.csr, san.conf, and ca.srl.

Run these commands to validate your work:

1. **Verify both certificates are ECDSA:**
   Bash
   openssl x509 -in ca.crt -text -noout | grep "Public Key Algorithm"
   openssl x509 -in public.crt -text -noout | grep "Public Key Algorithm"

   Expected for both: Public Key Algorithm: id-ecPublicKey
2. **Verify the chain of trust:**
   Bash
   openssl verify -CAfile ca.crt public.crt

   Expected Output: public.crt: OK

Need to convert private.key to PKCS#8

openssl pkcs8 -topk8 -nocrypt -in private.key -out private.pkcs8.key
mv private.key private.key.ec
mv private.pkcs8.key private.key