LO3 Demonstrate the use of cryptographic and cryptoanalysis tools for improving security in a virtual private network.

Illustration, using a diagram, encryption and decryption process functions in a PKI environment for a business scenario

3.1 Business scenario & how PKI solves it

In today's highly connected digital environment, any organisation dealing with highly private, sensitive information must have strong cryptographic infrastructures to ensure confidentiality, integrity, and authentication. Cyber Guard Solutions Pvt Ltd, and other highly recognised security organisations, are working with the likes of the Central Bank of Sri Lanka (CBSL) and Mount Elizabeth Hospital (Singapore), both of which are examples of organisations operating in environments that require transmission and exchange of sensitive, private data, secure identity assurance, and integrity data handling, which is relevant to both their organisational success and national interest. In these environments, Public Key Infrastructure (PKI) is the foundation for enabling trusted identity assurance and secure communication, in an environment handling tamper-proof data.

At the Central Bank of Sri Lanka, the key requirement is the secure exchange of financial data. The bank interacts with local commercial banks, international financial institutions, government ministries, and regulatory agencies. These interactions include processing inter-bank transactions, transmitting monetary policy data, and sending regulatory reporting documents. Without a PKI, these communications would be vulnerable to interception, tampering, or impersonation, which could undermine trust in the financial system. By implementing PKI, CBSL ensures that communication channels between itself and commercial banks are encrypted using TLS certificates, all participating institutions are authenticated via digital certificates, and sensitive financial documents are digitally signed to ensure their integrity. This guarantees that fraudulent instructions or altered data cannot be introduced without detection.

In parallel, Mount Elizabeth Hospital in Singapore presents another sensitive use case. The hospital relies on Cyber Guard’s Electronic Health Record (EHR) platform to manage patient information, prescriptions, and laboratory results. In this environment, confidentiality is paramount because patient data falls under strict privacy and regulatory requirements. Clinicians must access and update patient records from within the hospital network and through remote consultations, while external laboratories and consultants also interact with the system. PKI plays a vital role here by providing mutual authentication between users and the EHR system through client and server certificates, encrypting all data transfers with TLS to prevent eavesdropping, and enabling digital signatures for medical reports and prescriptions. This ensures that records cannot be tampered with and that the identity of the clinician who issued a prescription is provable in court or regulatory audits.

At the heart of PKI is the process of encryption and decryption, which finds confidentiality and protection from unauthorized users. In the case of asymmetric cryptography, all parties are issued with two cryptographic keys, a public and private key. For example, when CBSL sends inter-bank settlement instructions the message is encrypted using the receiver’s public key. Only the receiving bank can decrypt the message since it is the one with the matching private key. Therefore, it remains confidential, even if it is intercepted in transit. Just like in the healthcare example, when a clinician encrypts their sensitive patient data prior to transmission the recipient hospital server decrypts it with its private key. In practice PKI utilises asymmetric encryption along with symmetric algorithms in a hybrid encryption model: asymmetric keys are used to securely exchange a randomly generated symmetric key, generating symmetric encryption to encrypt most of the data using the faster AES-GCM. This is to provide efficiency and security, especially in environments that experience high volume, such as CBSL's inter-bank systems or Mount Elizabeth's patient management platform.

To illustrate how PKI secures communication, consider the use of Transport Layer Security (TLS) and mutual TLS (mTLS). When a clinician at Mount Elizabeth Hospital logs into the EHR system, the hospital server presents a digital certificate signed by Cyber Guard’s Certificate Authority (CA). The clinician’s device verifies this certificate by checking its chain of trust back to the trusted Root CA and confirming its validity using mechanisms such as the Online Certificate Status Protocol (OCSP). In the case of mutual TLS, the clinician’s device also presents its own certificate, which is verified by the server. This mutual exchange ensures that both parties are authenticated, eliminating the risks of phishing or identity spoofing. In CBSL’s environment, mTLS is especially critical for inter-bank communications, where both the central bank and commercial banks must mutually authenticate before exchanging sensitive transaction data. The TLS handshake uses PKI to negotiate ephemeral session keys, which are then used for symmetric encryption, protecting the confidentiality and integrity of financial and medical data in transit.

Another important Public Key Infrastructure (PKI) task for these scenarios involves digital signatures, which provide integrity and non-repudiation. For example, at the Central Bank of Sri Lanka (CBSL), senior public officials digitally sign regulatory reports and communications about monetary policy. They accomplish this by hashing the document, say using SHA-256, then they encrypt that hash using the signers private key to produce the digital signature. Any recipient can verify the authenticity of the document by decrypting the signature using the signers public key and compare the hash with the one that they create. If the hash values are the same, the document is authentic and has not changed. Similarly, in the case of Mount Elizabeth Hospital, clinicians will digitally sign discharge summaries and prescriptions. Audit systems in the hospital are then able to demonstrate that a particular clinician issued a prescription, thereby preventing denial (non-repudiation) and accountability. This gives legal evidence of a chain of custody, which is necessary in both regulated financial environments, and for compliance in the healthcare environment.

The enrolment and certificate issuance process are also crucial in both contexts. At CBSL, when a new commercial bank is onboarded into the national payment infrastructure, the Registration Authority (RA) verifies the bank's identity and forwards its Certificate Signing Request (CSR) to the Certificate Authority. Once approved, the CA issues a digital certificate binding the bank's cryptographic key pair to its verified identity. This certificate is then published in a central repository accessible to all participating institutions. Similarly, at Mount Elizabeth Hospital, a new clinician is required to provide professional credentials and undergo identity checks before being issued a digital certificate. The private key is securely stored on the clinician's device or on a hardware token, while the certificate is distributed for verification during authentication processes. This process guarantees that only authorised individuals or organisations receive certificates, preventing impersonation or unauthorised access.

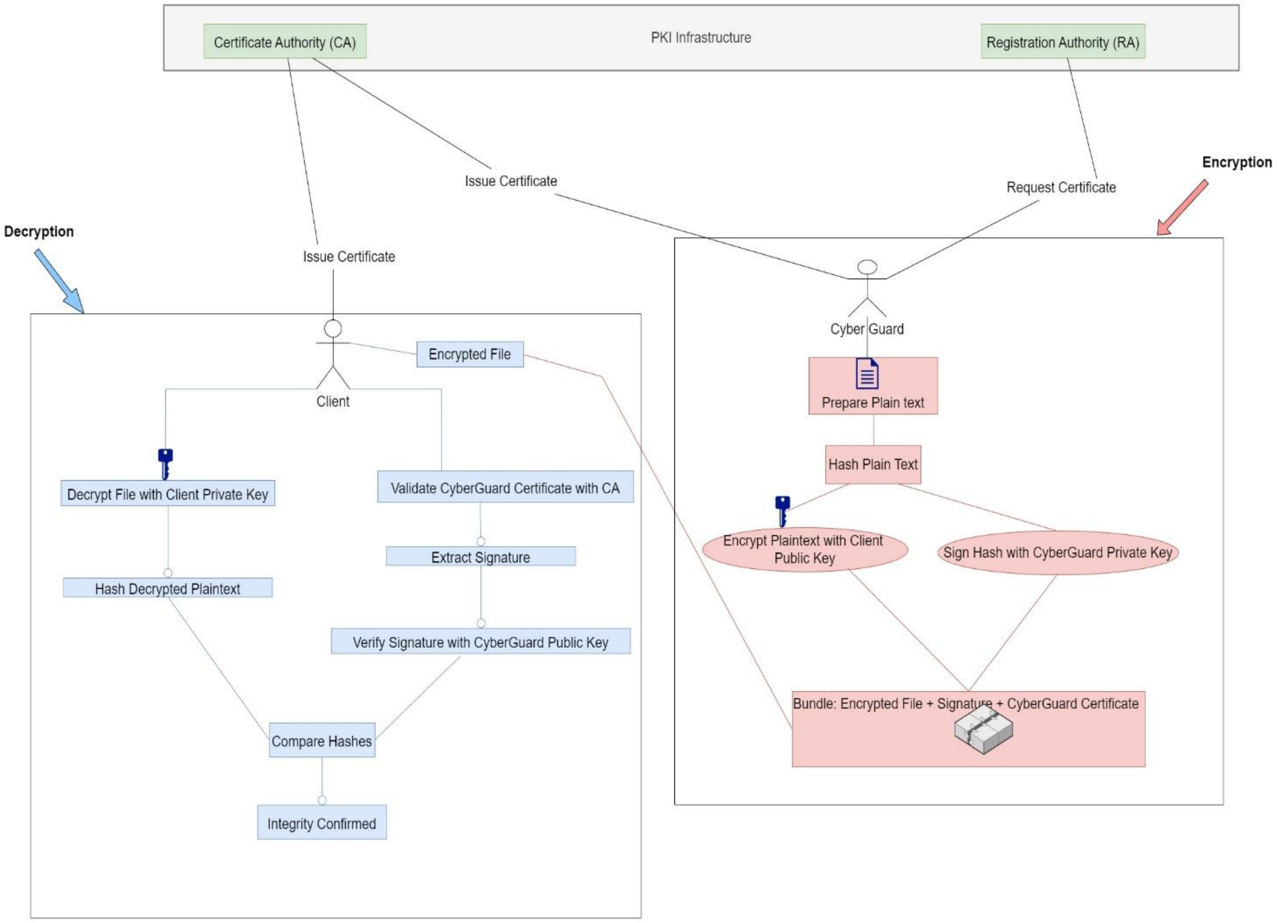
The successful implementation of PKI across both CBSL and Mount Elizabeth Hospital demonstrates how the technology can address three critical needs: secure communication, authentication, and data integrity. Encryption and decryption functions safeguard the confidentiality of sensitive data such as financial transactions and patient records. TLS and mutual TLS provide strong mutual authentication between systems and users, ensuring that only authorised entities participate in data exchanges. Digital signatures guarantee data integrity and non-repudiation, preventing tampering and enabling accountability. Furthermore, certificate lifecycle management (issuance, renewal, and revocation) ensure that compromised or expired certificates are quickly invalidated, maintaining continuous trust within the PKI ecosystem.

Ultimately, the use of PKI at both the Central Bank of Sri Lanka and Mount Elizabeth Hospital highlights its versatility and effectiveness in different industries. While the bank prioritises the security of financial transactions and inter-bank communications, the hospital focuses on protecting patient confidentiality and ensuring the authenticity of clinical records. In both cases, PKI provides a scalable, auditable, and legally enforceable infrastructure that strengthens trust, prevents fraud, and ensures compliance with international standards. This demonstrates the indispensable role of PKI in protecting sensitive data and identities in modern business environments.

Illustrations to demonstrate the PKI components and process flows encryption & decryption

The diagram illustrates the complete Public Key Encryption (PKE) process within a PKI infrastructure, covering both encryption and decryption stages. CyberGuard requests and obtains a certificate from the Certificate Authority (CA) via the Registration Authority (RA), ensuring its identity is verified. In the encryption stage, CyberGuard prepares and hashes the plaintext, encrypts it using the client’s public key, and signs the hash with its own private key. The encrypted file, digital signature, and CyberGuard’s certificate are bundled together and sent to the client. During decryption, the client validates CyberGuard’s certificate with the CA, verifies the signature using CyberGuard’s public key, and decrypts the ciphertext with its private key. Finally, the client compares the hash of the decrypted plaintext with the signed hash to confirm integrity. This process ensures confidentiality, integrity, authenticity, and non-repudiation of the transmitted data.

This Public Key Encryption (PKE) process demonstrates how CyberGuard securely exchanges information with clients using a PKI framework. The encryption stage guarantees confidentiality by using the client's public key, while the digital signature from CyberGuard ensures authenticity and non-repudiation. On the client side, certificate validation and hash comparison confirm data integrity and trust in CyberGuard's identity. Together, these steps provide a complete end-to-end security model against tampering and impersonation.


Certificate Authority (CA)
PKI Infrastructure
Registration Authority (RA)
Issue Certificate
Request Certificate
Encryption
Decryption
Issue Certificate
Cyber Guard
Client
Encrypted File
Prepare Plain text
Hash Plain Text
Decrypt File with Client Private Key
Validate CyberGuard Certificate with CA
Extract Signature
Encrypt Plaintext with Client Public Key
Sign Hash with CyberGuard Private Key
Hash Decrypted Plaintext
Verify Signature with CyberGuard Public Key
Bundle: Encrypted File + Signature + CyberGuard Certificate
Compare Hashes
Integrity Confirmed

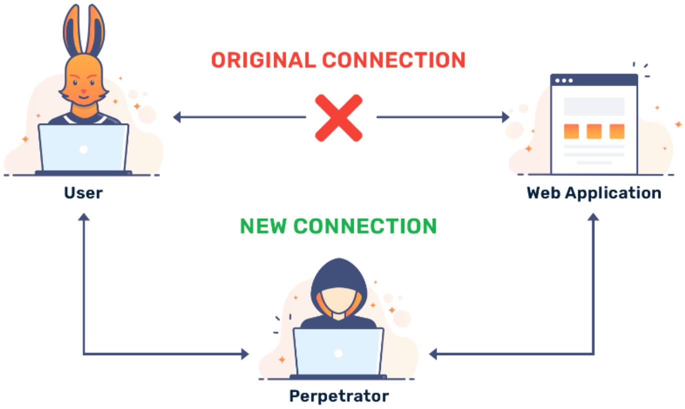
Designing of a security case, representative of a business scenario, to solve a security threat

3.2 Security Issue - Man-in-the-Middle (MITM) Attack

Business Scenario

Consider a consultant from the Central Bank of Sri Lanka (CBSL) who is required to upload a settlement file while travelling. The consultant connects to an airport Wi-Fi network that appears legitimate but is in fact a rogue access point operated by an attacker. Because the consultant's laptop does not enforce certificate pinning and no VPN has been initiated, the attacker transparently proxies the connection between the consultant and the CBSL transaction server. In this position, the attacker is able to intercept the uploaded file, alter the beneficiary account number, and forward the modified transaction to the bank's server. From the server's perspective the request appears genuine, as it comes through an apparently valid session. This leads to a fraudulent transfer and undermines the integrity of the settlement process. A parallel situation can occur at Mount Elizabeth Hospital, where a clinician accessing patient records over public Wi-Fi risks exposure of confidential data to the same class of attack. Both cases highlight how insecure communication channels open the door for devastating breaches affecting confidentiality, integrity, and availability of critical services.

What is MITM Attack?



A man-in-the-middle (MITM) attack is a cyber-attack in which a threat actor puts themselves in the middle of two parties, typically a user and an application, to intercept their communications and data exchanges and use them for malicious purposes like making unauthorized purchases or hacking.

By secretly standing between the user and a trusted system, such as a website or application, a cybercriminal can easily obtain sensitive data. The user assumes they're interacting exclusively with a trustworthy site and willingly relinquishes login credentials, financial information, or other compromising data. (Magnusson, 2024)

How a MITM Attack Happens

A Man-in-the-Middle (MITM) attack begins when an adversary positions themselves on the network path between two communicating parties so they can observe and manipulate traffic in real time. Common methods for gaining that on-path position include rogue Wi-Fi access points that mimic legitimate hotspots, ARP/DNS spoofing on local networks, compromised routers, or malicious proxies that force victims to route traffic through the attacker's machine. Once on-path the attacker intercepts initial protocol handshakes (for example the TLS Client Hello) and may attempt active proxying terminating the client's connection locally and opening a separate connection to the legitimate server, thereby creating two sessions the attacker controls. If endpoints fail to properly validate server identity (for instance by ignoring certificate warnings or accepting self-signed certificates) the attacker can impersonate the server and decrypt, modify, or replay messages; where robust protocols such as TLS 1.3 are correctly deployed the handshake itself is integrity-protected, but client misconfiguration or weak authentication can still permit successful proxying. The upshot is that, without strict identity verification and ephemeral session keys, sensitive artefacts such as credentials, financial instructions, or patient records can be read or altered in transit without either endpoint being immediately aware. (Rescorla, 2018)

What the Attacker Does as a Company Representative

From a corporate perspective the threat is amplified when the attacker either compromises or impersonates an insider for example a consultant, contractor, or employee who legitimately has access to client systems. In such a scenario the attacker may deploy a manipulated VPN client, configure a rogue access point that re-advertises the company SSID, or persuade colleagues to accept a certificate warning to establish a proxy session. Once traffic flows through that proxy the attacker can harvest long-lived credentials, capture session tokens, modify files (for example changing beneficiary details in a settlement batch) or inject secondary payloads such as remote access malware. For organisations operating in regulated spaces central banking and healthcare the consequences of a single successful MITM incident are severe: fraudulent transfers, unauthorized disclosure of protected health information, regulatory penalties, and reputational damage that undermines client trust. The corporate reality is that human error (clicking through warnings), legacy services that lack modern TLS configurations, and insufficient endpoint hardening remain frequent enablers for these attacks.

How PKE, KEM and DEM Mitigate the Problem

A layered cryptographic approach combining Public Key Encryption (PKE) with robust PKI controls, Key Encapsulation Mechanisms (KEM) for ephemeral key exchange, and Data Encapsulation Mechanisms (DEM) for high-performance authenticated encryption effectively neutralizes the principal MITM vectors.

Public Key Encryption (PKE) implemented via a well-managed PKI binds identity to keys and prevents impersonation when clients strictly validate certificate chains, perform OCSP/CRL checks or use OCSP stapling, and adopt certificate pinning or mutual TLS (mTLS) for high-risk channels. These measures stop attackers from presenting forged certificates unless they control a trusted CA or the private signing key; keeping CA root keys in HSMs and issuing short-lived leaf certificates reduces that exposure. (Anon, Ietf.org. , 2025)

Key Encapsulation Mechanisms (KEM) protect session key establishment by enabling ephemeral, efficient asymmetric encapsulation of a short symmetric key. When sessions use ephemeral KEMs (or ephemeral Diffie-Hellman) the session keys have forward secrecy: even if an attacker later obtains long-term private keys, they cannot decrypt previously captured sessions. For long-term resilience, hybrid KEM constructions (classical + post-quantum algorithms such as CRYSTALS-Kyber standardized by NIST) are recommended to guard against “harvest now, decrypt later” attacks. NIST guidance and draft SP-800-227 provide recommended usage patterns for safely deploying KEMs. (Alagic, G., 2025)

Data Encapsulation Mechanisms (DEM) use authenticated encryption with associated data (AEAD) ciphers (for example AES-GCM, ChaCha20-Poly1305, or nonce-misuse-resistant variants like AES-GCM-SIV where appropriate) to encrypt bulk payloads using the session key established by the KEM. AEAD ensures both confidentiality and integrity: any attempt by a MITM to alter ciphertext will be detected because tag verification fails on the recipient. Operationally, DEMs require strict nonce/IV management, automatic key rotation, and use of authenticated modes for all sensitive payloads to prevent replay or forgery. (S. Gueron, 2019)

When applied together, PKE prevents identity spoofing, KEMs secure the session key exchange and provide forward secrecy, and DEMs ensure payload integrity and confidentiality creating a cryptographic chain that makes passive eavesdropping, active proxying, and undetected tampering infeasible. For maximal protection in high-risk environments, combine these with mTLS for administrative links, HSM custody of master keys and root CAs, strict TLS 1.3 configurations, enforced VPN/machine posture checks for remote clients, and continuous monitoring of certificate transparency and OCSP/CRL status

Also, mitigation can be done using VPN's,

- Two major types of VPN technologies are suitable for Cyber Guard's client environments:

IPsec VPN (Internet Protocol Security): Operates at the network layer of the OSI model. Provides end-to-end encryption of IP packets between the remote user and the internal system. Uses cryptographic protocols such as Encapsulating Security Payload (ESP) and Authentication Header (AH) to guarantee confidentiality, integrity, and authentication. Ideal for CBSL, where high-volume, system-to-system communications between banks require strong, low-level encryption to protect financial data.

SSL VPN (Secure Sockets Layer): Operates at the application layer and relies on TLS/SSL protocols. Provides secure access through a standard web browser, without requiring complex client software. Ideal for Mount Elizabeth Hospital, where doctors and consultants may require quick, browser-based access to patient portals from diverse devices. Ensures that even if accessed through an untrusted network, all communication between the doctor's device and the hospital's servers remains encrypted. By deploying VPNs, Cyber Guard ensures that all remote access traffic is encrypted, even if transmitted over untrusted public Wi-Fi networks or the general internet. Additionally, VPNs provide mutual authentication, ensuring that only verified users with the correct credentials and certificates can connect. This safeguards confidentiality and integrity, while protecting clients from eavesdropping, identity spoofing, and data leakage.

Cyber Guard Solutions Pvt Ltd has employees located across the globe and clients within tightly regulated industries, including central banking and healthcare sectors. During the pandemic and in its effort to improve global to serve its clients, with frequent travel necessitating remote access to processes and systems, Cyber Guard employed a distributed workforce in situations where many staff and consultants need to access sensitive back-end systems of their clients remotely from home and on the road. They were prepared for this scenario, however, this picture creates a high-risk situation for these users as it places them within an environment where the requirement for unencrypted traffic over public networks would expose information, whether they are credentials, transaction instructions, or patient records to interception. Applying security terminology, this type of event raises all of the classic "CIA" risk loss of confidentiality (attackers were able to read sensitive data) integrity (attackers were to alter transactions or records), and availability (interruption of service as a result of hijacked sessions). To that end, the case study will establish a need for a solution that will allow Central Bank of Sri Lanka (CBSL) employees and partner banks the opportunity to exchange interbank financial data and making electronic health records (EHR) accessible remotely for Mount Elizabeth Hospital (Singapore) clinicians, from outside the headline of their secure premises.

The threat landscape involves man-in-the-middle (MITM) attacks on public Wi-Fi, packet sniffing on untrusted networks, credential harvesting using phishing or keyloggers, and session hijacking of poorly protected web portals. The vulnerabilities include reliance on password-only authentication, some internal services not providing end-to-end encryption, and very little oversight on remote logins. The impact could be grave: for CBSL it could be fraudulent transfers or disruption of the national payments system; for Mount Elizabeth it could be unauthorized disclosure of patient records, which violates HIPAA/PDPA obligations and causes reputational and legal damage.

To mitigate these risks, Cyber Guard will deploy enterprise-grade VPN solutions to encrypt all remote traffic, combined with strong authentication and continuous monitoring. For CBSL, an IPsec VPN operating at the network layer will be implemented. This protocol encapsulates and encrypts IP packets themselves using Encapsulating Security Payload (ESP) and Authentication Header (AH), providing confidentiality, integrity and peer authentication (Kent & Seo, 2005). IPsec supports site-to-site tunnels with high throughput, which is essential for interbank traffic that may include bulk transaction files or automated settlement streams. For Mount Elizabeth Hospital, a browser-based SSL/TLS VPN is more appropriate. This approach uses the existing TLS stack to create an encrypted tunnel between the clinician's device and the hospital portal without requiring heavy client software, thus facilitating quick and secure access from diverse endpoints. (Oppliger, 2016)

Supporting controls include:

- Multi-factor authentication (MFA), preferably certificate-based smart cards or app-based one-time codes, to reduce the impact of stolen credentials.

- Periodic key and certificate renewal through an internal Public Key Infrastructure (PKI) to ensure cryptographic freshness.

- Centralised logging and anomaly detection of VPN activity to spot unusual login patterns.

- Integration with a Hardware Security Module (HSM) or secure enclave for cryptographic key storage.

This security case illustrates how VPNs can be tailored to different operational contexts high-volume, low-latency IPsec tunnels for interbank links, and flexible SSL/TLS VPNs for clinician access while maintaining a unified security posture. By aligning cryptographic controls with specific business processes, Cyber Guard can deliver both operational continuity and regulatory compliance for its most sensitive clients. (Stallings, 2017)

Assessing Security Risks and Challenges of Using Cloud-Hosted PKI in a Private Network (M3)

Although Virtual Private Networks (VPNs) address the immediate need to encrypt traffic over insecure channels, they do not in themselves solve the problem of trust and authentication. Every VPN tunnel depends on certificates and keys issued by a Public Key Infrastructure (PKI). In Cyber Guard's case, certificates are used to authenticate thousands of CBSL staff, partner banks and Mount Elizabeth clinicians, as well as to secure machine-to-machine traffic between critical systems. Managing those certificates securely is therefore as important as encrypting the traffic.

Traditionally, this meant running a PKI entirely on-premises. However, many organisations are now migrating to cloud-hosted PKI services because they reduce the operational burden of maintaining Certificate Authorities (CAs), issuing and revoking certificates, and integrating with cloud-native applications. In Cyber Guard's environment this might mean that certificate issuance and key management for CBSL and Mount Elizabeth clinicians is performed by a third-party provider.

While this model can offer scalability, elasticity and cost savings, it also introduces a new category of risk that has to be examined critically. The following sections discuss these risks in detail and propose mitigation strategies, drawing on recognised guidance from NIST. (ENISA, 2021)

Trust and Control

Moving PKI services to the cloud moves a critical trust anchor - the CA - out of the organisation's direct control. In CBSL's interbank systems which PKI is used to authenticate high value transactions, any compromise, insider threat or configuration error at a cloud provider, undermines sovereign national financial security controls. Even issuing a valid certificate to an unauthorised entity could unwittingly lead to a man-in-the-middle (MITM) attack against CBSL's payment channels with disastrous impact. Guidance in the sector e.g., NIST SP 800-57 (2020), recommends organisations maintain their root CA features physically and logically controlled within the organisation, and only use intermediate CAs at a cloud service provider. This creates a trust anchor that is only under Cyber Guard/CBSL's control and allows the scalability of certificate issuance to the edge.

Latency and Availability

PKI operations occur at every TLS handshake, VPN connection, and digital signature verification. If the cloud-hosted PKI becomes unavailable or experiences high latency, thousands of authentications could fail simultaneously, creating a denial-of-service event that halts financial transactions or blocks hospital clinicians from accessing patient records in emergencies. According to ENISA (2021), critical PKI services must achieve near-continuous availability through geographic redundancy and robust Service Level Agreements (SLAs). Cyber Guard should also implement mitigations such as local caching of Certificate Revocation Lists (CRLs) and the use of OCSP stapling to reduce live lookups, ensuring that VPN clients can continue to validate certificates even if the upstream PKI service is temporarily unreachable. (ENISA, 2021)

Key Management Risks

The most sensitive component of any PKI is the private key material: both the CA’s signing keys, and any private keys present within the issued certificates. If these are stored in regular VMs or databases in the cloud, they may be visible to privileged insiders or susceptible to external attack. Best practice, as laid out in NIST’s FIPS 140-3 standard, is to store all private key materials in Hardware Security Modules (HSMs) with tamper-resistant hardware, strict access controls, and audit logging of every key operation. For CBSL, a compromised CA private key could authorize bogus interbank messages; for Mount Elizabeth, it could sign counterfeit prescriptions or access tokens, thus endangering patient safety and legal compliance. Cyber Guard must accordingly exercise due diligence to ensure that an outsourced cloud PKI provides transparency to its use of FIPS-certified HSMs, and its key-management practices.

Compliance and Jurisdictional Issues

Cloud providers often replicate data and keys across multiple data centres in different jurisdictions. For CBSL, this raises data sovereignty concerns under Sri Lankan financial regulations. For Mount Elizabeth Hospital, it may violate Singapore's Personal Data Protection Act (PDPA) if patient-related certificate data are stored overseas. These legal risks are not hypothetical data-protection authorities have imposed fines on organisations that moved personal data to unapproved regions. Cyber Guard must therefore verify the provider's data-residency guarantees and, where necessary, choose a region-specific or government-accredited provider. In highly sensitive cases, Cyber Guard may opt for a "hybrid" PKI model where only non-sensitive operations are cloud-hosted, and all sensitive root keys remain on-premises. (ENISA, 2021)

Mitigation Strategies and Best Practices

An effective mitigation plan allows Cyber Guard to weigh the obvious advantages of cloud-hosted public key infrastructure (PKI) including the benefits of centralized certificate management, automated enrolment and revocation, and integration with cloud-native applications against the risks it brings. Rather than seeing PKI as strictly a binary yes-or-no decision, Cyber Guard can implement layered risk mitigation and be deliberate about integrating multiple controls (technical, contractual, and organizational) to uphold the trust source within the cryptographic environment.

Hybrid PKI design - A useful control is to implement a hybrid PKI architecture where the high-assurance root certificate authority (CA) is kept on-premises and controlled directly by Cyber Guard or CBSL, and only the intermediate or issuing CAs are in the cloud. This way, the most sensitive keys are maintained under Cyber Guard's sovereignty and possibly managed in an on-premises data centre with physical security and strict perimeter controls, while Cyber Guard is still able to enjoy the scalability and automation features of a cloud PKI for day-to-day certificate issuance. NIST SP 800-57 (2020) explicitly states "the trust anchor should remain under the organization's control in order to avoid third-party compromise of the entire chain of trust**."**

HSM enforcement - All private keys both root and subordinate must be generated, stored and used only within Hardware Security Modules (HSMs) that are FIPS 140-3 certified. HSMs are tamper-resistant cryptographic devices that protect keys from exfiltration, even from privileged insiders. By requiring HSM use in contracts with the cloud provider, Cyber Guard ensures that no key material is found in plaintext on a standard virtual machine or database, preventing its theft and misuse. This approach also conforms to best practice payment industry and healthcare cryptography.

Strict SLAs and audit rights – Cyber Guard will archive its security requirements technically but will also need to incorporate its contractually binding expectations of security through Service Level Agreements (SLAs) to cover not just uptime, but incident response times, breach reporting obligations, change-management processes and the right to initiate independent security audits of the cloud provider's PKI ecosystem. These clauses will provide Cyber Guard leverage for strengthening security processes and assurance that the cloud provider adheres to relevant regulations throughout the service lifecycle.

Security tests - Even with Service Level Agreements in place, Cyber Guard should still routinely pen test and assess the security of the cloud PKI's interfaces, administrative portals and APIs. Then it can identify potential weaknesses, such as misconfigured access controls or weak API authentication, before they can be exploited. Additionally, with the documentation of regular testing, it would provide some evidence that would support Cyber Guard's due diligence to its regulators and clients. Testing the PKI can also be supplemented by code reviews, configuration audits, and red-team tests of the PKI environment.

Certificate pinning and transparency logs. On the client side, Cyber Guard can deploy additional safeguards such as certificate pinning and Certificate Transparency (CT) logs. Certificate pinning ensures that VPN clients and internal applications only trust certificates signed by expected CAs, making it harder for an attacker with a rogue certificate to succeed. CT logs, meanwhile, provide a public, append-only record of all certificates issued, enabling Cyber Guard to monitor for unexpected or unauthorised issuance and to react quickly if it occurs. (Oppliger, 2016)

Ongoing oversight – Cyber Guard should implement ongoing oversight and anomaly detection of certificate usage patterns. For instance, sudden spikes in certificate issuance, unplanned certificate revocations, or unstated authentication attempts may signify compromise. Automated alerts and dashboards provide the security team with early detection of potential breaches and the opportunity to act before a breach becomes problematic.

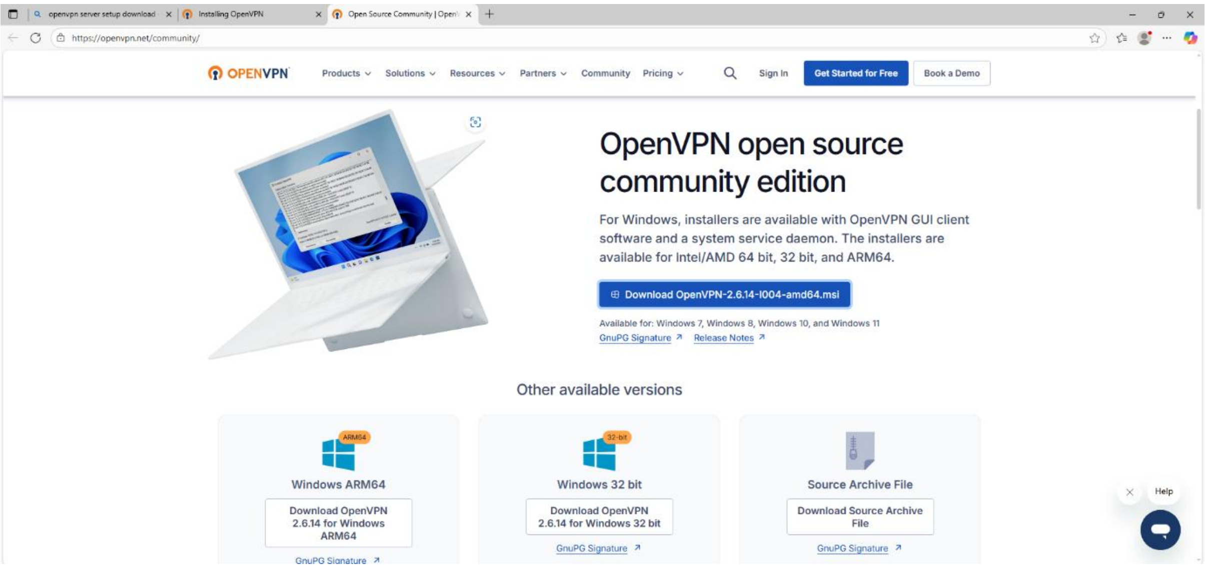
Taken together, these steps take the layered approach necessary to protect the trustworthiness gifted by VPN authentication and encryption - at the same time as Cyber Guard updates its hardware and software infrastructures. Together, these steps replicate ENISA's (2021) "Cloud Security for PKI" recommendations and NIST's key-management framework. Following these steps, Cyber Guard will be able to demonstrate due diligence to its regulators, provide reassurance to its high-value customers such as CBSL and Mount Elizabeth Hospital, and keep its cryptographic foundations secure, resilient, and compliant.

Implement the system designed, in response to a security case, using cryptographic and cryptanalysis methods or tools.

To put the proposed security solution into practice, a secure Virtual Private Network (VPN) server was deployed for Cyber Guard Solutions Pvt Ltd using the OpenVPN platform. OpenVPN was chosen because it is an industry-standard, open-source solution that supports strong cryptographic protocols such as AES-256-GCM and TLS 1.3, certificate-based authentication and HMAC integrity checks. The implementation was designed to create an encrypted tunnel for remote CBSL staff and Mount Elizabeth Hospital clinicians, configure robust access controls, and verify secure connectivity from a client device. The following subsections document the environment used, the installation and configuration steps, the security features enabled and provide screenshots and command logs as evidence of a successful VPN deployment.

Step 1 – Selecting the Proper Setup

In this step the correct OpenVPN setup package was selected and downloaded from the official OpenVPN website. Using an official, signed installer ensures that the software has not been tampered with and that the cryptographic binaries are genuine. Choosing the proper version for the operating system also ensures compatibility with the EasyRSA toolset used to build the PKI.



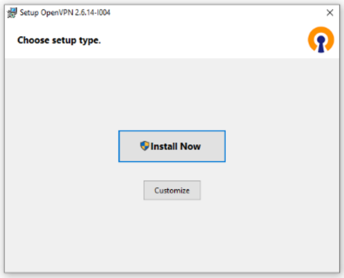
OpenVPN open source community edition

For Windows, installers are available with OpenVPN GUI client software and a system service daemon. The installers are available for Intel/AMD 64 bit, 32 bit, and ARM64.

Available for: Windows 7, Windows 8, Windows 10, and Windows 11

Step 2 – Installing of OpenVPN

The OpenVPN installer was run on the server machine. This installs the OpenVPN service, its configuration files, and the EasyRSA scripts used to create keys and certificates. During installation administrative privileges are required so that the service can register network drivers (TAP adapters) and add firewall rules.

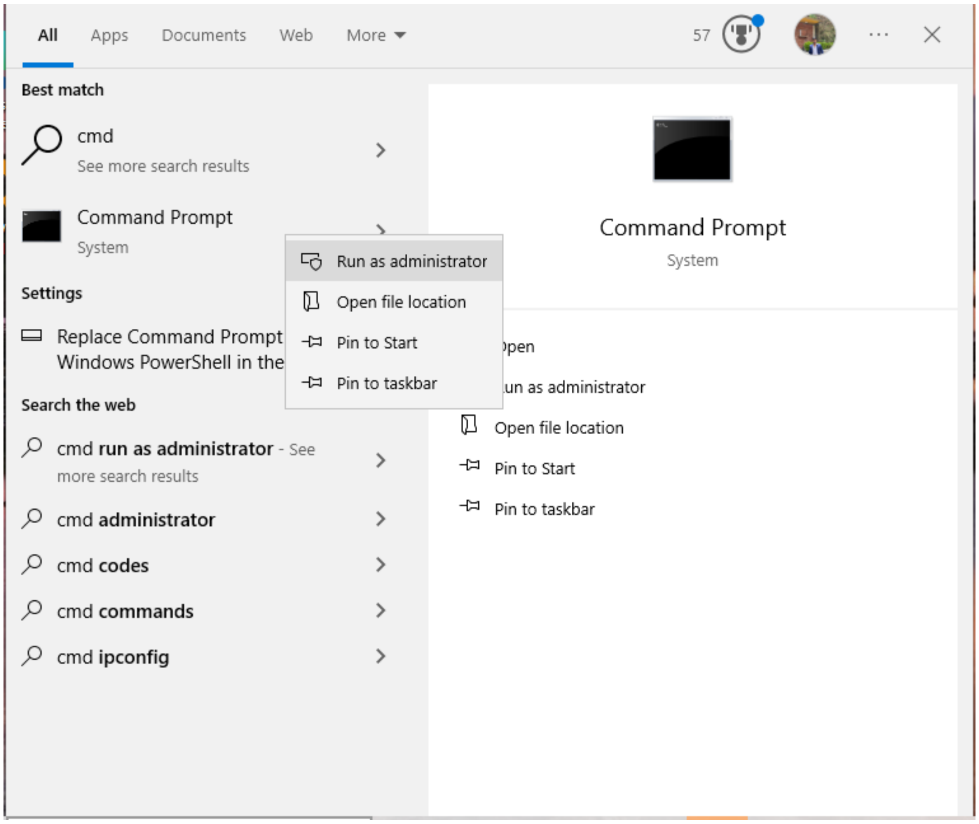


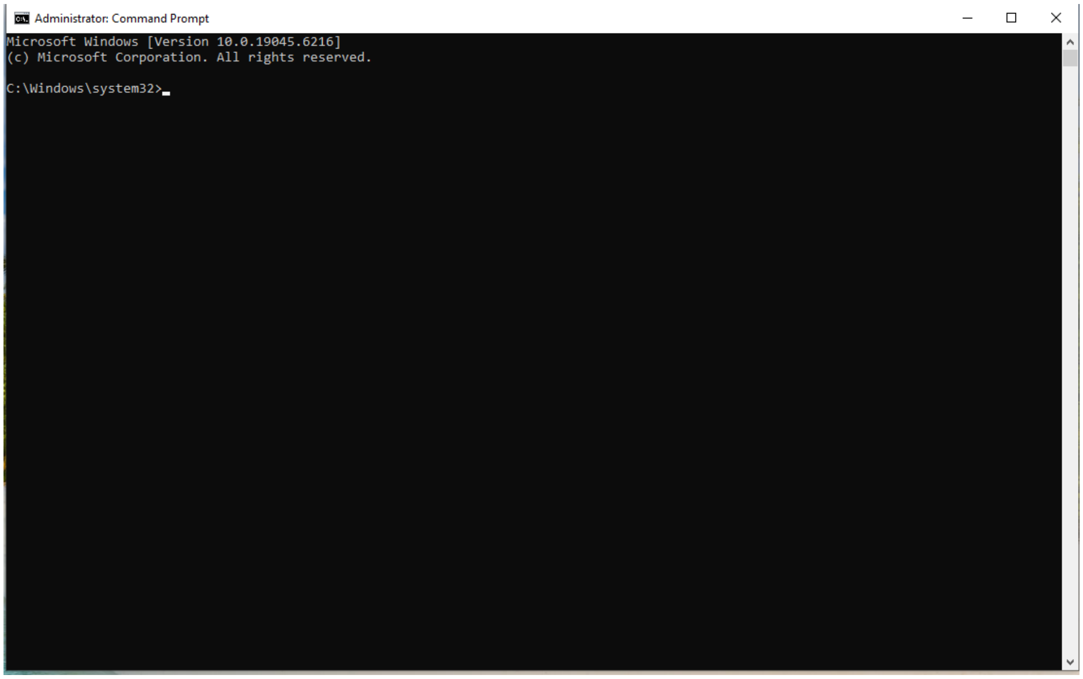
Setup OpenVPN 2.6.14-I004

Choose setup type.

Step 3 – Running CMD as administrator

A Command Prompt was opened with “Run as Administrator”. Many of the EasyRSA and OpenVPN setup commands need elevated permissions to write keys and configuration files into protected directories and to register services with Windows. Running as administrator ensures the commands execute successfully.

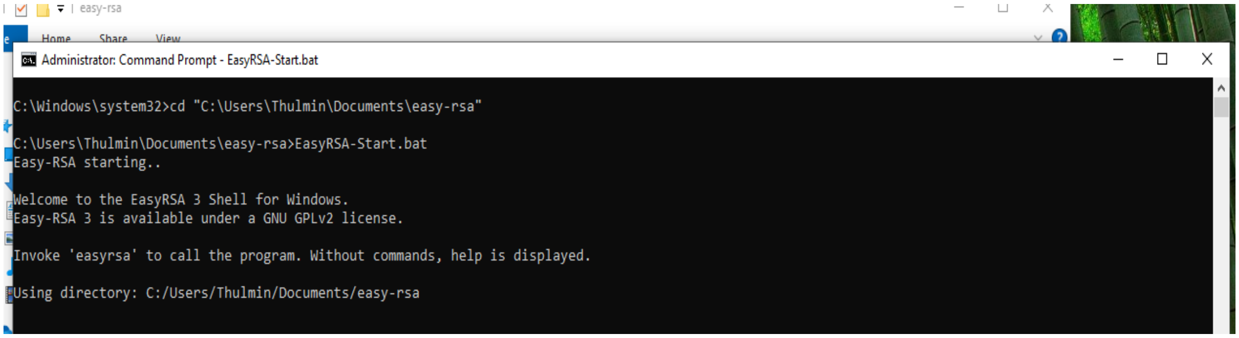




```
Microsoft Windows [Version 10.0.19045.6216]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>_
```

Step 4 – Navigating to EasyRSA folder

The working directory was changed to the EasyRSA folder included with the OpenVPN installation. EasyRSA is a simple Certificate Authority (CA) management tool bundled with OpenVPN. All subsequent commands for creating the PKI are run from this location

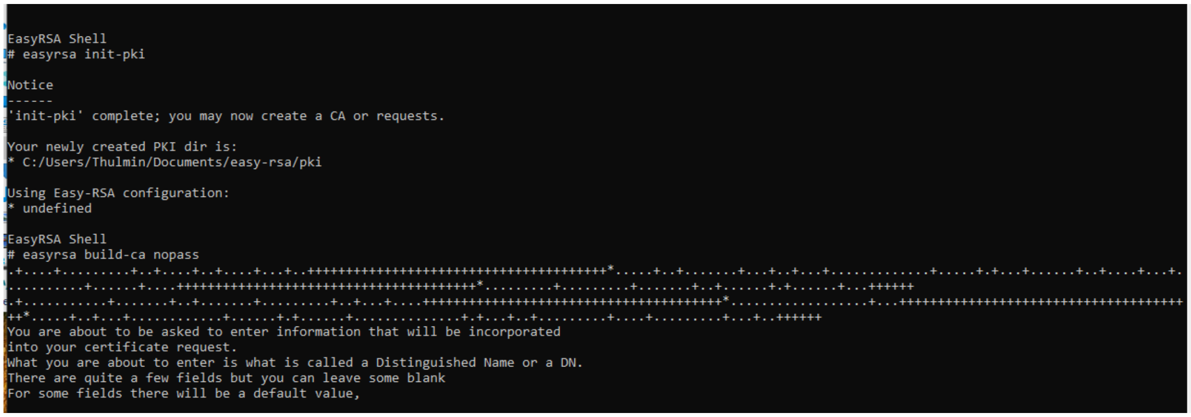


```
C:\Windows\system32>cd "C:\Users\Thulmin\Documents\easy-rsa"

C:\Users\Thulmin\Documents\easy-rsa>EasyRSA-Start.bat
Easy-RSA starting..

Welcome to the EasyRSA 3 Shell for Windows.
Easy-RSA 3 is available under a GNU GPLv2 license.

Invoke 'easyrsa' to call the program. Without commands, help is displayed.

Using directory: C:/Users/Thulmin/Documents/easy-rsa
```

Step 5 – Run EasyRSA commands

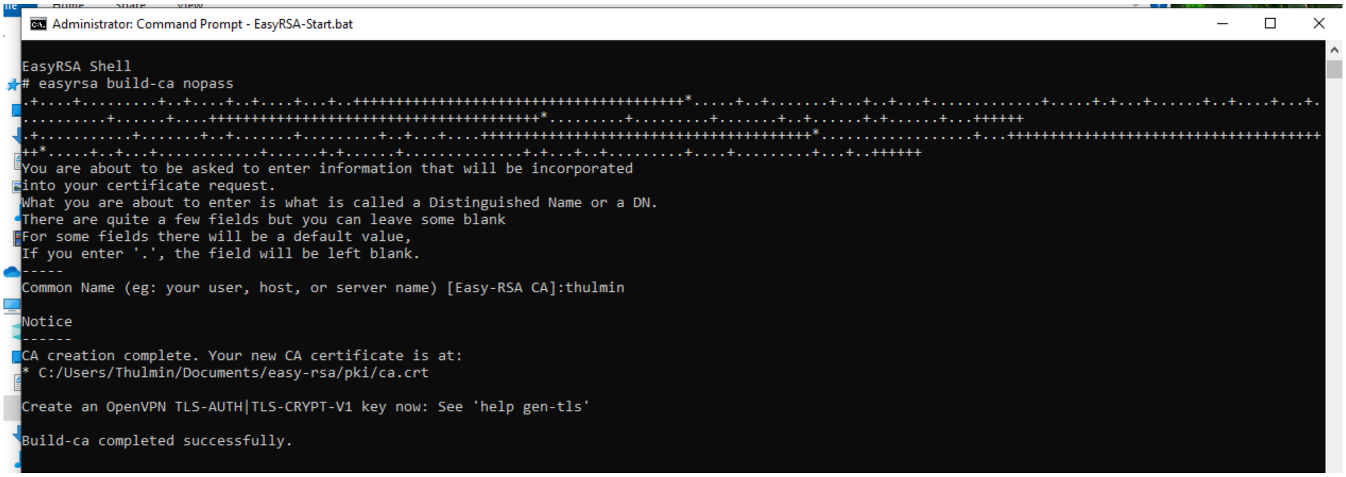
At this stage the foundation of the VPN’s trust infrastructure was created. EasyRSA is a lightweight Certificate Authority (CA) management tool bundled with OpenVPN that automates the process of generating and signing certificates. The command easyrsa init-pki initialises a new Public Key Infrastructure (PKI) directory structure, creating the folders where all keys and certificates will be stored. Without this step there is no secure location to manage cryptographic material. The command easyrsa build-ca nopass then generates the CA’s private key and a self-signed root certificate. This root CA is the “trust anchor” for the entire VPN ecosystem: every server and client certificate will be signed by it, and OpenVPN clients will be configured to trust this CA. In a production deployment the CA private key would normally be encrypted with a passphrase and stored offline, but for a classroom or lab exercise the nopass option is acceptable to simplify testing.

- easyrsa init-pki



```
EasyRSA Shell
# easyrsa init-pki

Notice
------
'init-pki' complete; you may now create a CA or requests.

Your newly created PKI dir is:
* C:/Users/Thulmin/Documents/easy-rsa/pki

Using Easy-RSA configuration:
* undefined

EasyRSA Shell
# easyrsa build-ca nopass
.+....+........+..+....+..+....+...+..+++++++++++++++++++++++++++++++++++++++*.....+..+.......+...+..+...+............+.....+.+...+......+..+....+...+.
..........+......+...+++++++++++++++++++++++++++++++++++++++*.........+.........+.......+..+......+.+......+...++++++
.+...........+.......+..+.......+.........+..+...+...+++++++++++++++++++++++++++++++++++++++*.................+...++++++++++++++++++++++++++++++++++++
++*.....+..+...+...........+......+.+......+.............+.+...+..+.........+....+.........+...+..++++++
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

- easyrsa build-ca nopass

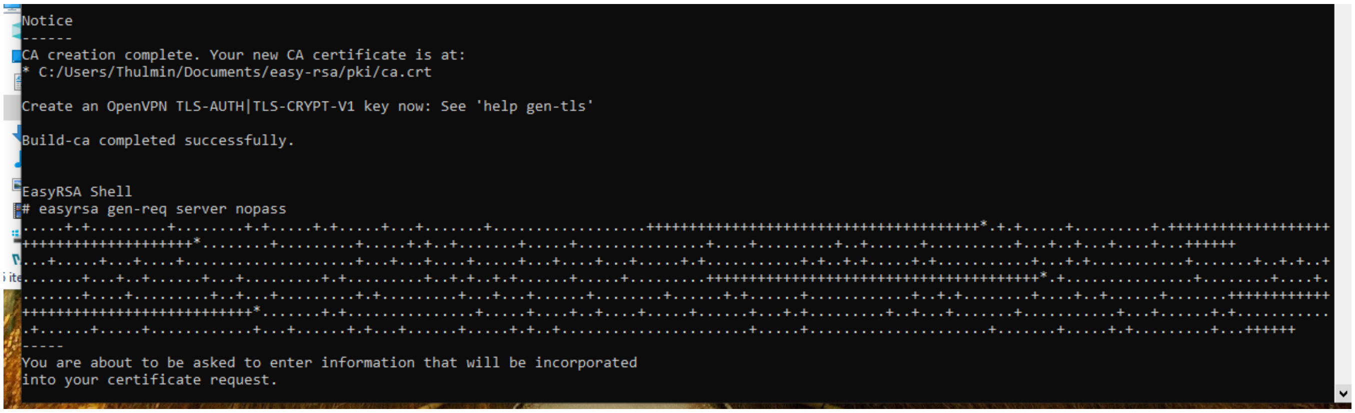


```
EasyRSA Shell
# easyrsa build-ca nopass
.+....+........+..+....+..+....+...+..+++++++++++++++++++++++++++++++++++++++*.....+..+.......+...+..+...+............+.....+.+...+......+..+....+...+.
..........+......+...+++++++++++++++++++++++++++++++++++++++*.........+.........+.......+..+......+.+......+...++++++
.+...........+.......+..+.......+.........+..+...+...+++++++++++++++++++++++++++++++++++++++*.................+...++++++++++++++++++++++++++++++++++++
++*.....+..+...+...........+......+.+......+.............+.+...+..+.........+....+.........+...+..++++++
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:thulmin

Notice
------
CA creation complete. Your new CA certificate is at:
* C:/Users/Thulmin/Documents/easy-rsa/pki/ca.crt

Create an OpenVPN TLS-AUTH|TLS-CRYPT-V1 key now: See 'help gen-tls'

Build-ca completed successfully.
```

Step 6 – Generating server certificate key

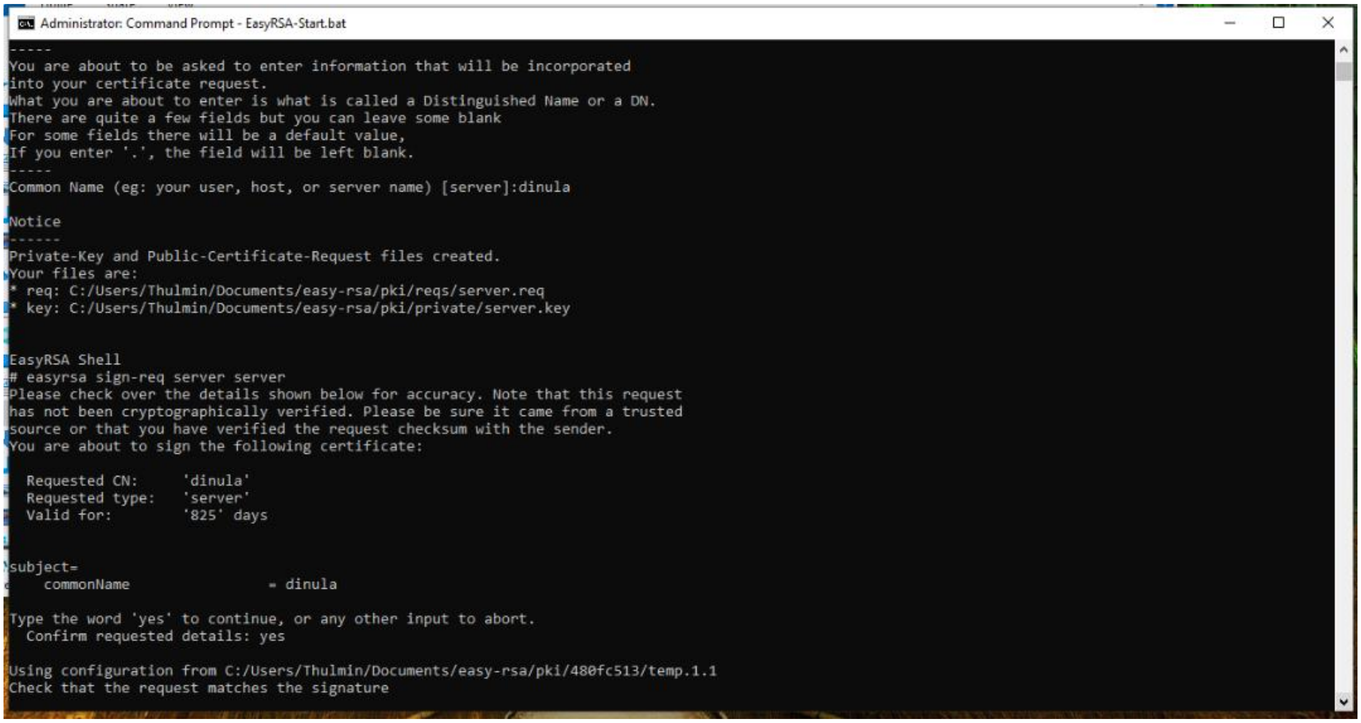
Once the CA exists, the OpenVPN server itself needs its own identity in the form of a key pair and certificate. The command easyrsa gen-req server nopass generates a private key and a certificate signing request (CSR) for the server. This CSR contains the server’s public key and identity information but no signature. The next command easyrsa sign-req server server uses the CA created in Step 5 to sign the CSR, producing a server certificate that is trusted by the CA. This two-step process mirrors how certificates are issued on the public internet. Having a CA-signed server certificate means that when clients connect they can verify they are talking to the genuine OpenVPN server, preventing man-in-the-middle attacks and ensuring that only a properly authenticated server can establish tunnels.

- easyrsa gen-req server nopass



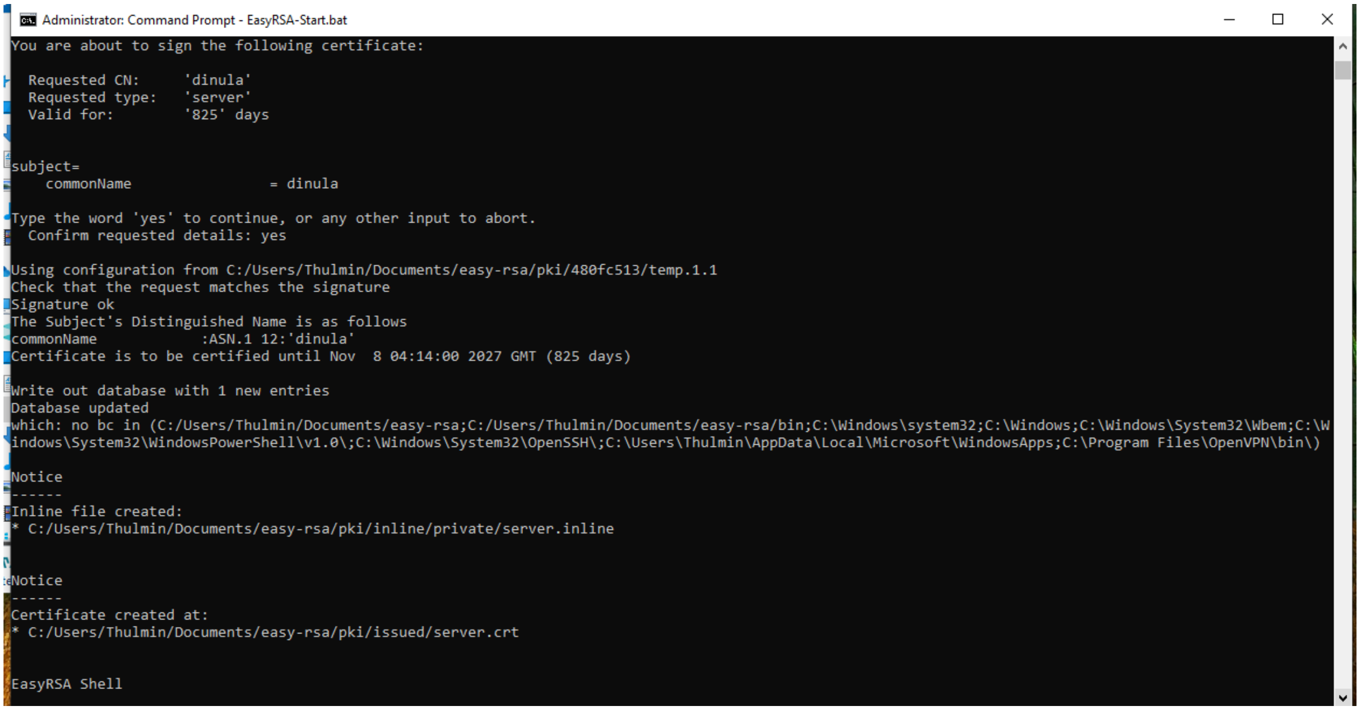
```
Notice
------
CA creation complete. Your new CA certificate is at:
* C:/Users/Thulmin/Documents/easy-rsa/pki/ca.crt

Create an OpenVPN TLS-AUTH|TLS-CRYPT-V1 key now: See 'help gen-tls'

Build-ca completed successfully.


EasyRSA Shell
# easyrsa gen-req server nopass
.....+.+.........+........+.+.....+.+.....+...+.......+.................++++++++++++++++++++++++++++++++++++++++*.+.+.....+.........+.++++++++++++++++++++
++++++++++++++++++++*........+.........+.....+.+..+.......+.....+...............+....+..........+..+......+..........+...+..+...+....+...++++++
...+.....+...+....+...................+...+...+....+.....+...+....+...+.....+.+...........+.+..+.+.....+.+...........+...+.+...........+.......+..+.+..+
.......+...+..+......+...+.........+.+.........+..+.+..+.+.....+.....+.........+++++++++++++++++++++++++++++++++++++++++*.+...............+.........+....+.
.......+....+.........+..+...+.........+.+.........+...+...+......+.........+......+.+......+............+..+.........+....+..+......+.......++++++++++++
+++++++++++++++++++++++++++*.......+.+...............+.....+....+..+....+.....+.....+...+.+.........+..+...+.......+...........+...+......+.+...........
.+.......+.....+.............+...+......+.+...+......+.....+.+..+.....................+.....+.....................+.......+.....+.+.........+...++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

- easyrsa sign-req server server



Administrator: Command Prompt - EasyRSA-Start.bat

```
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [server]:dinula

Notice
------
Private-Key and Public-Certificate-Request files created.
Your files are:
* req: C:/Users/Thulmin/Documents/easy-rsa/pki/reqs/server.req
* key: C:/Users/Thulmin/Documents/easy-rsa/pki/private/server.key


EasyRSA Shell
# easyrsa sign-req server server
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.
You are about to sign the following certificate:

  Requested CN:     'dinula'
  Requested type:   'server'
  Valid for:        '825' days


subject=
    commonName                = dinula

Type the word 'yes' to continue, or any other input to abort.
  Confirm requested details: yes

Using configuration from C:/Users/Thulmin/Documents/easy-rsa/pki/480fc513/temp.1.1
Check that the request matches the signature
```

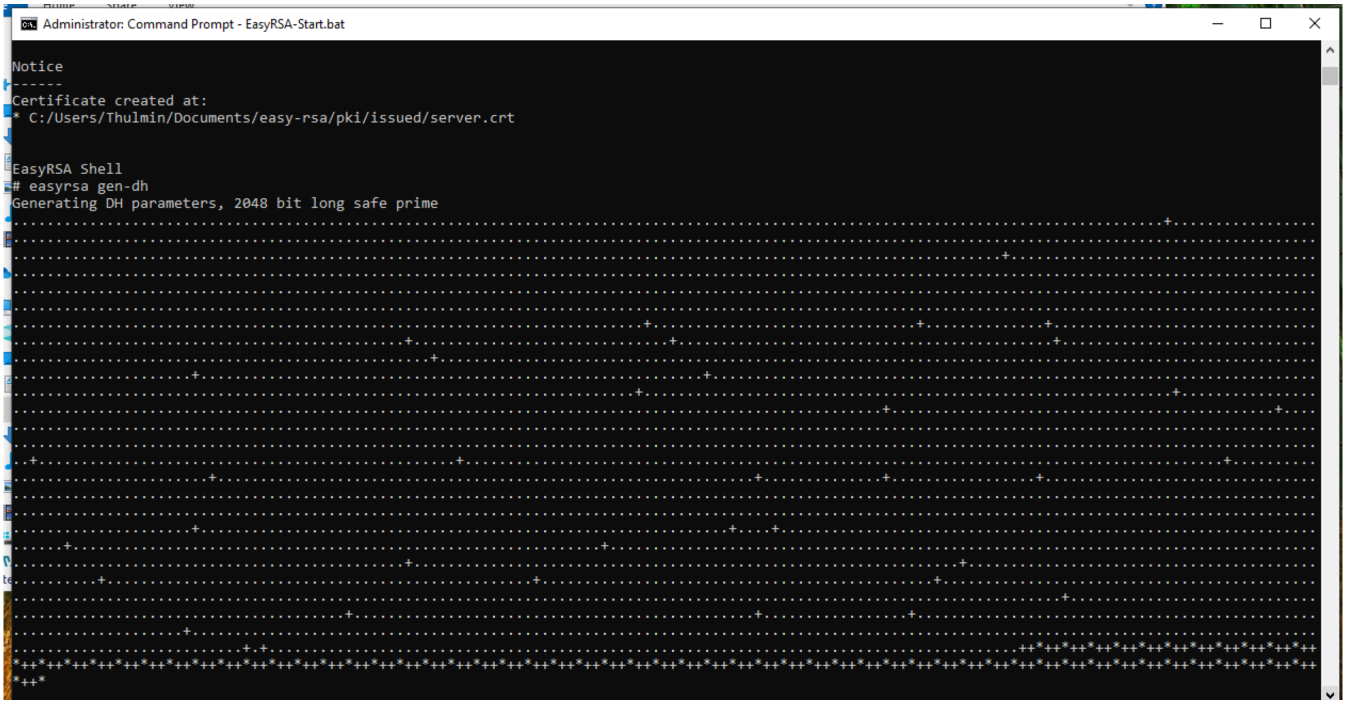


```
You are about to sign the following certificate:

  Requested CN:     'dinula'
  Requested type:   'server'
  Valid for:        '825' days


subject=
    commonName                = dinula

Type the word 'yes' to continue, or any other input to abort.
  Confirm requested details: yes

Using configuration from C:/Users/Thulmin/Documents/easy-rsa/pki/480fc513/temp.1.1
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName            :ASN.1 12:'dinula'
Certificate is to be certified until Nov  8 04:14:00 2027 GMT (825 days)

Write out database with 1 new entries
Database updated
which: no bc in (C:/Users/Thulmin/Documents/easy-rsa;C:/Users/Thulmin/Documents/easy-rsa/bin;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\W
indows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Users\Thulmin\AppData\Local\Microsoft\WindowsApps;C:\Program Files\OpenVPN\bin\)

Notice
------
Inline file created:
* C:/Users/Thulmin/Documents/easy-rsa/pki/inline/private/server.inline


Notice
------
Certificate created at:
* C:/Users/Thulmin/Documents/easy-rsa/pki/issued/server.crt


EasyRSA Shell
```

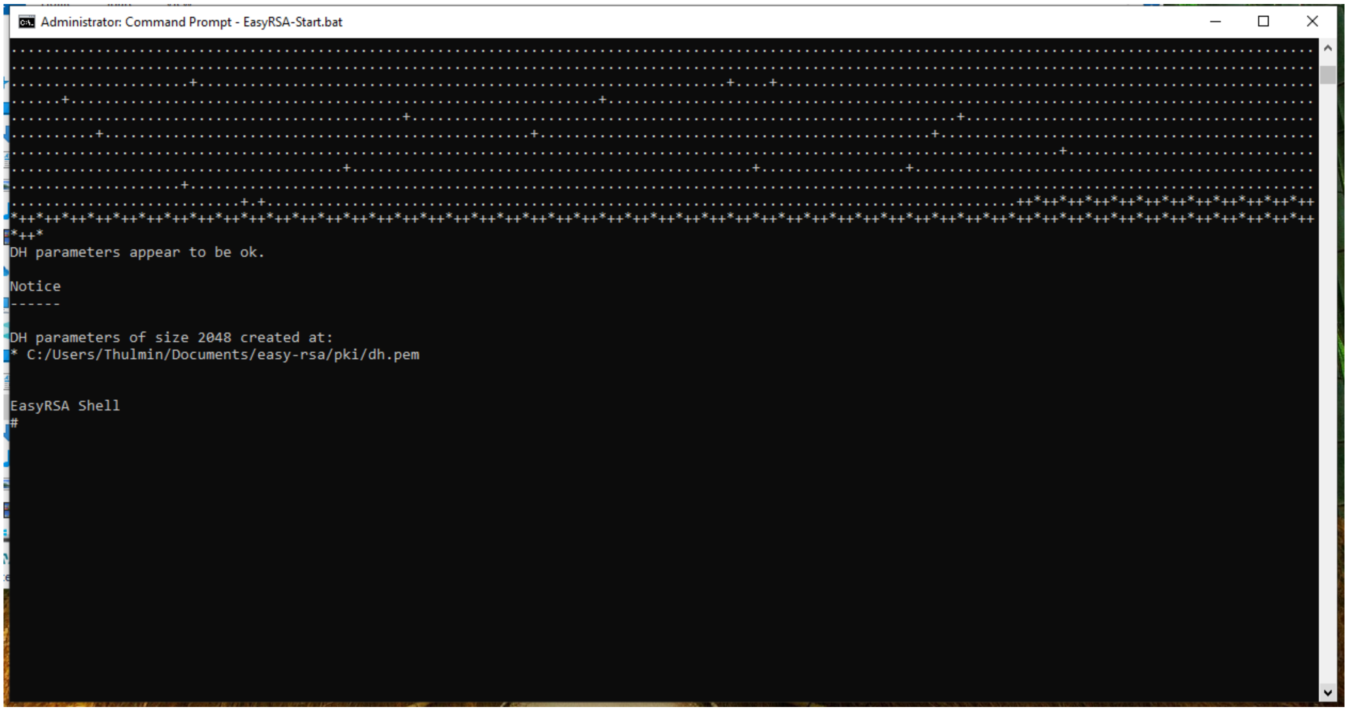
Step 7 – Generating Diffie-Hellman parameters

Diffie–Hellman (DH) parameters are required for the server and clients to perform secure key exchange during the TLS handshake. The command easyrsa gen-dh generates a fresh set of DH parameters. This allows the server and client to agree on a unique shared session key over an untrusted network without sending the key itself. Using freshly generated DH parameters strengthens forward secrecy: even if a long-term key were compromised later, past VPN sessions would remain confidential. This step is crucial to the cryptographic strength of the VPN tunnel and is explicitly recommended in OpenVPN's security documentation.

- easyrsa gen-dh



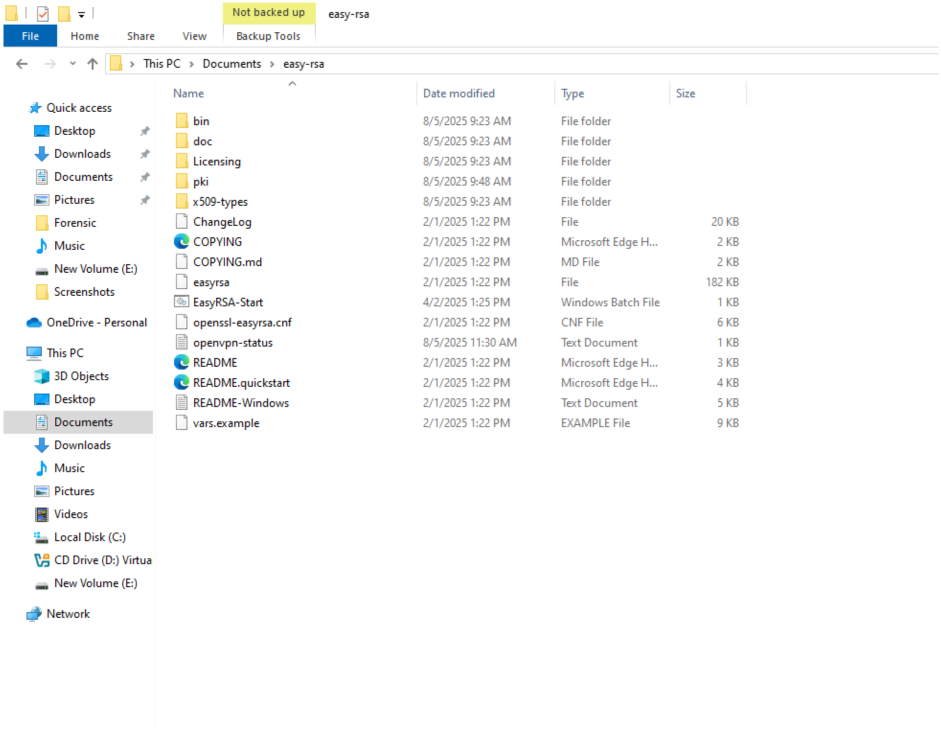
```
Notice
------
Certificate created at:
* C:/Users/Thulmin/Documents/easy-rsa/pki/issued/server.crt


EasyRSA Shell
# easyrsa gen-dh
Generating DH parameters, 2048 bit long safe prime
.......................................................................................................................................+.................
.........................................................................................................................................................
....................................................................................................................+....................................
.........................................................................................................................................................
.........................................................................................................................................................
.........................................................................................................................................................
..........................................................................+...............................+..............+...............................
..............................................+..............................+............................................+..............................
.................................................+.......................................................................................................
.....................+...........................................................+.......................................................................
.........................................................................+..............................................................+................
......................................................................................................+.............................................+....
.........................................................................................................................................................
.........................................................................................................................................................
..+.................................................+.........................................................................................+..........
.......................+...............................................................+..............+.................+................................
.........................................................................................................................................................
.........................................................................................................................................................
.....................+..............................................................+....+...............................................................
......+..............................................................+...................................................................................
..............................................+................................................................+.........................................
..........+..................................................+..............................................+............................................
...........................................................................................................................+.............................
.......................................+...............................................+.................+...............................................
....................+....................................................................................................................................
...........................+.+........................................................................................++*++*++*++*++*++*++*++*++*++*++*++
*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++
*++*
```



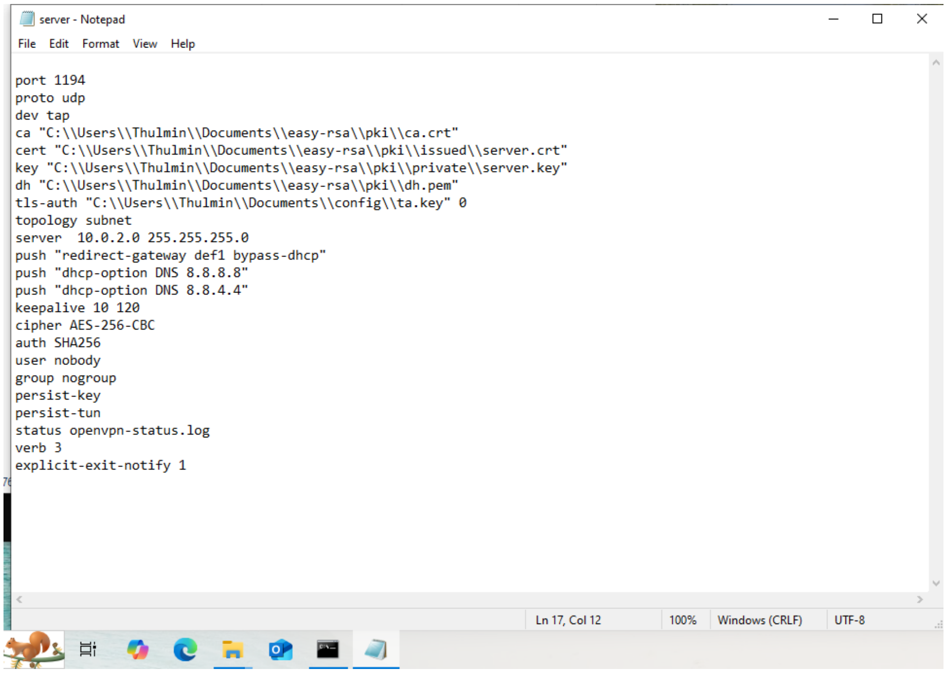
```
.........................................................................................................................................................
.........................................................................................................................................................
.....................+..............................................................+....+...............................................................
......+..............................................................+...................................................................................
..............................................+................................................................+.........................................
..........+..................................................+..............................................+............................................
...........................................................................................................................+.............................
.......................................+...............................................+.................+...............................................
....................+....................................................................................................................................
...........................+.+........................................................................................++*++*++*++*++*++*++*++*++*++*++*++
*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++*++
*++*
DH parameters appear to be ok.

Notice
------

DH parameters of size 2048 created at:
* C:/Users/Thulmin/Documents/easy-rsa/pki/dh.pem


EasyRSA Shell
#
```

Step 8 – Save Location

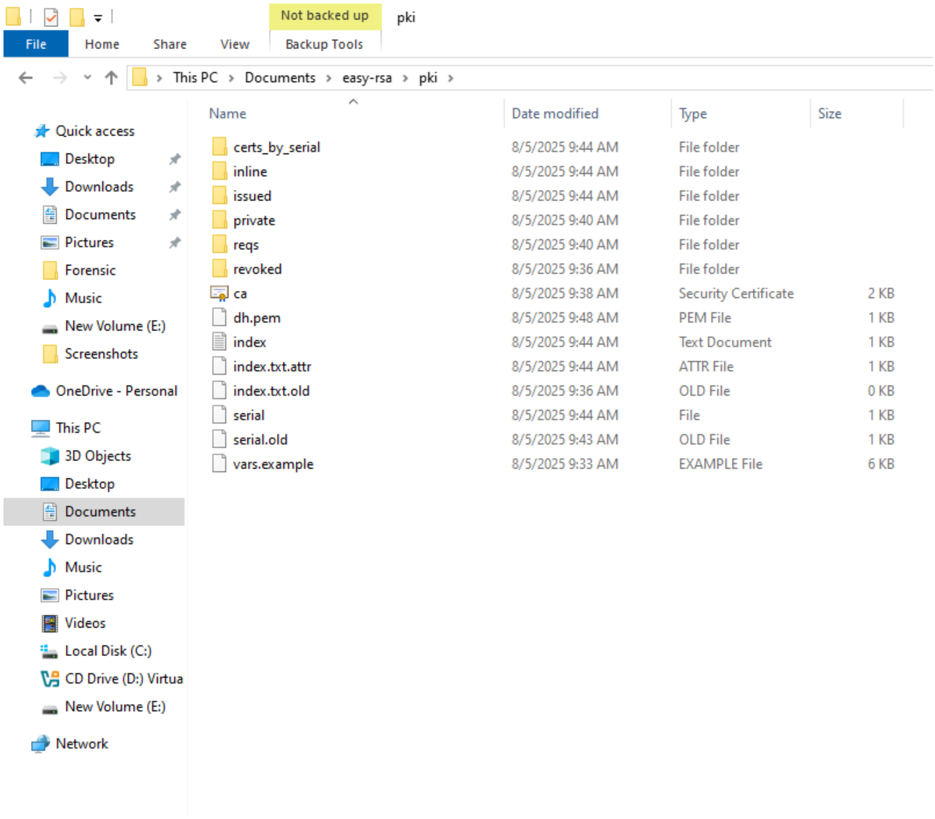
After the CA, server certificate and DH parameters have been generated, they must be placed in the correct directories for the OpenVPN service to reference them. In this step all the generated .crt, .key and .pem files were copied into the server's OpenVPN "config" or "keys" folders. Correct file placement ensures that the server configuration file can point to the right paths for ca, cert, key and dh directives. This organisational step may seem administrative but is vital to avoid misconfiguration, which is one of the most common causes of VPN security weaknesses.



Name	Date modified	Type	Size
bin	8/5/2025 9:23 AM	File folder	
doc	8/5/2025 9:23 AM	File folder	
Licensing	8/5/2025 9:23 AM	File folder	
pki	8/5/2025 9:48 AM	File folder	
x509-types	8/5/2025 9:23 AM	File folder	
ChangeLog	2/1/2025 1:22 PM	File	20 KB
COPYING	2/1/2025 1:22 PM	Microsoft Edge H...	2 KB
COPYING.md	2/1/2025 1:22 PM	MD File	2 KB
easyrsa	2/1/2025 1:22 PM	File	182 KB
EasyRSA-Start	4/2/2025 1:25 PM	Windows Batch File	1 KB
openssl-easyrsa.cnf	2/1/2025 1:22 PM	CNF File	6 KB
openvpn-status	8/5/2025 11:30 AM	Text Document	1 KB
README	2/1/2025 1:22 PM	Microsoft Edge H...	3 KB
README.quickstart	2/1/2025 1:22 PM	Microsoft Edge H...	4 KB
README-Windows	2/1/2025 1:22 PM	Text Document	5 KB
vars.example	2/1/2025 1:22 PM	EXAMPLE File	9 KB



```
port 1194
proto udp
dev tap
ca "C:\\Users\\Thulmin\\Documents\\easy-rsa\\pki\\ca.crt"
cert "C:\\Users\\Thulmin\\Documents\\easy-rsa\\pki\\issued\\server.crt"
key "C:\\Users\\Thulmin\\Documents\\easy-rsa\\pki\\private\\server.key"
dh "C:\\Users\\Thulmin\\Documents\\easy-rsa\\pki\\dh.pem"
tls-auth "C:\\Users\\Thulmin\\Documents\\config\\ta.key" 0
topology subnet
server  10.0.2.0 255.255.255.0
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"
keepalive 10 120
cipher AES-256-CBC
auth SHA256
user nobody
group nogroup
persist-key
persist-tun
status openvpn-status.log
verb 3
explicit-exit-notify 1
```



Name	Date modified	Type	Size
certs_by_serial	8/5/2025 9:44 AM	File folder	
inline	8/5/2025 9:44 AM	File folder	
issued	8/5/2025 9:44 AM	File folder	
private	8/5/2025 9:40 AM	File folder	
reqs	8/5/2025 9:40 AM	File folder	
revoked	8/5/2025 9:36 AM	File folder	
ca	8/5/2025 9:38 AM	Security Certificate	2 KB
dh.pem	8/5/2025 9:48 AM	PEM File	1 KB
index	8/5/2025 9:44 AM	Text Document	1 KB
index.txt.attr	8/5/2025 9:44 AM	ATTR File	1 KB
index.txt.old	8/5/2025 9:36 AM	OLD File	0 KB
serial	8/5/2025 9:44 AM	File	1 KB
serial.old	8/5/2025 9:43 AM	OLD File	1 KB
vars.example	8/5/2025 9:33 AM	EXAMPLE File	6 KB

Step 9 – Final Output sequence completed

This step confirmed that all cryptographic materials had been created successfully and were ready to be used by the OpenVPN service. The console output at this stage acts as evidence that the PKI build completed without errors. With the CA certificate, server certificate, server private key and DH parameters in place, the OpenVPN service can now be started. Once the service is running and the client configuration files have been exported, remote users can import their profiles and establish fully encrypted tunnels to the server. In other words, this final step is where the designed security case becomes a working implementation, demonstrating to stakeholders that Cyber Guard’s VPN infrastructure can protect sensitive financial and healthcare data over insecure networks.