# gRPC Security Notes

P4 Control Plane uses a *secure-by-default* approach. The northbound interface for infrap4d are secured by mTLS.

Detailed info about gRPC Ports and Certificate Management is available in another wiki page here: gRPC Ports & Certificate Management

This document covers details of cryptography and versions/algorithms for reference.

## gRPC version

History of gRPC versions integrated in infrap4d

| MEV-TS version | IPDK version | gRPC version | Notes & Comments |
|---|---|---|---|
| 0.71 | | 1.42.0 | |
| | IPDK 22.07 | | |
| 0.8, 0.9, 0.9.8 | | 1.50.1 | |
| | IPDK 23.01 | | |
| 1.0, 1.1 | | 1.54.2 | |
| | IPDK 23.07 | | |
| 1.2 to 2.0 | | 1.59.2 | |
| | IPDK 24.01 | | |

Further details of various CVEs addressed with each upgrade is available on this page: Upgrade History of Dependent Libraries

## Crypto Library

gRPC integration in infrap4d is built with OpenSSL instead of the default BoringSSL.

All crypto implementation thus come from OpenSSL

The version of OpenSSL libraries depend on user environment & OS version. infrap4d will build with both OpenSSL 1.1.1x and 3.x. Users are highly encouraged to move to OpenSSL 3.x as 1.1.1x version went EOL in Sep 2023.

## TLS Version

The gRPC library supports both TLS versions 1.2 and 1.3.

Depending on the gRPC client establishing connection with infrap4d, it will default to the highest version mutually supported. Example: if gRPC client also supports TLS versions 1.2 and 1.3, it will default to TLS 1.3

## Cipher Suites

Asymmetric algorithms are used for setting up and maintaining the channel (endpoint authentication, certificate verification etc).

Algorithms used for key and certificate generation include RSA, SHA512. See: https://github.com/ipdk-io/stratum-dev/blob/split-arch/tools/tls/generate-certs.sh

For bulk transfer once channel is established, symmetric algorithms are used.

The set of cipher suites supported by gRPC are: https://github.com/grpc/grpc/blob/b517a3d792c340b9d9c97c979e0bc52635d038e5/src/core/lib/config/config_vars.yaml#L122C1-L122C1

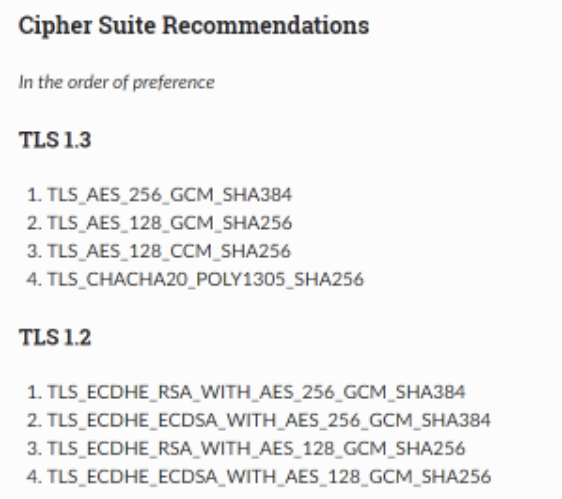
```
default: "TLS_AES_128_GCM_SHA256:\
    TLS_AES_256_GCM_SHA384:\
    TLS_CHACHA20_POLY1305_SHA256:\
    ECDHE-ECDSA-AES128-GCM-SHA256:\
    ECDHE-ECDSA-AES256-GCM-SHA384:\
    ECDHE-RSA-AES128-GCM-SHA256:\
    ECDHE-RSA-AES256-GCM-SHA384"
```

For TLS 1.3, only the following three cipher suites are considered safe:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

The cipher suite negotiation occurs during the TLS handshake between the client and the server. The negotiation process involves both the client and server presenting a list of supported cipher suites in order of preference. The final selection is based on the mutual agreement between the client and the server, with consideration given to security, compatibility, and available algorithms.

Cipher suite recommendations according to Intel Crypto Guidelines (v1.6.5) are: https://readthedocs.intel.com/cryptoteam/tls_guidelines/tls_recommendations.html



**Cipher Suite Recommendations**

*In the order of preference*

**TLS 1.3**

1. TLS_AES_256_GCM_SHA384
2. TLS_AES_128_GCM_SHA256
3. TLS_AES_128_CCM_SHA256
4. TLS_CHACHA20_POLY1305_SHA256

**TLS 1.2**

1. TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
2. TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
3. TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
4. TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

As of MEV-TS 1.1, infrap4d does not make any explicit calls to force select a cipher suite. If required in future, the following OpenSSL API call can be used to select: SSL_CTX_set_ciphersuites()

It has been observed during testing with IPsec Recipe and strongSwan gRPC client, the default cipher suite selected is **TLS_AES_128_GCM_SHA256**

## References:

- Intel Crypto guidelines: https://goto.intel.com/cryptoguidelines