PV Access TLS

- A Technical Proposal for TLS in PV Access

Osprey DCS

George McIntyre

EPICS Collaboration Meeting, Fermilab, 2023

TLS for PV Access
Agenda

- What is planned?
- Features
  - Server Authentication
  - Encryption
  - Server Certificate rotation
  - Compression
  - Client Authentication
  - Authorization
- How it works?
  - The low level details

What is Planned?

Current Work – commissioned by SLAC - 2023

PV Access TLS Feature Implementation:

George McIntyre

- TLS Channel Search over TCP
- TLS Handshake
- TLS Encapsulation with Encryption and Signature
- Support for Server Certificate & Rotation
- Support for Compression
- Command line tool support – ***pvput, pvget, pvmonitor***, …
- Unit test suite

EPICS Technical Security Analysis:

Michael Davidsaver

- Report
- Roadmap

Java implementation – *maintainer Kay Kasemir*

- https://github.com/ControlSystemStudio/phoebus/tree/master/core/pva

C++ implementation – *maintainer Michael Davidsaver*

- https://mdavidsaver.github.io/pvxs

Documentation

- https://github.com/epics-base/pvAccessCPP/wiki/protocol

Out of scope

Features

- Client Certificates
- Client configuration mappings for TLS parameters
- Add TLS to Channel Access
- UDP Broadcast search
- UDP response
- Beacon messages
- Add additional TLS beacon messages for servers supporting both TLS and TCP
- Any changes to support TLS in Gateways
- Any changes to support TLS in EPICS Python (pvaPy)
- Any changes to support TLS in PV Database

Repositories

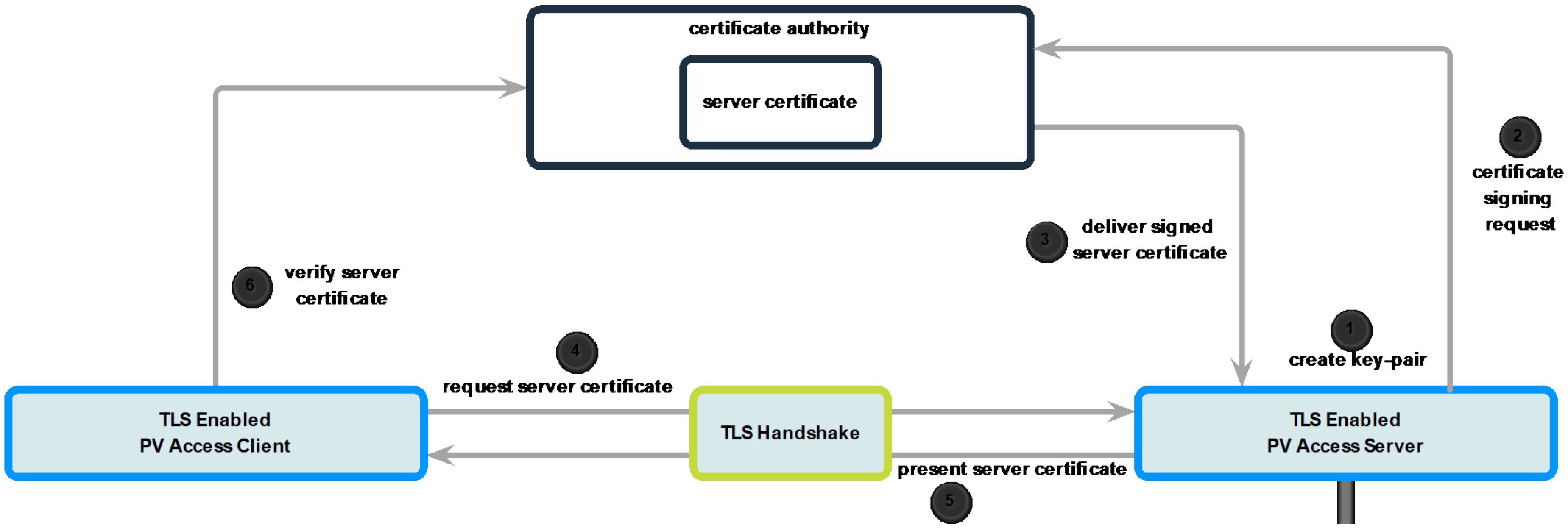
EPICS base Java

- https://github.com/epics-base/epicsCoreJava
- https://github.com/epics-base/pvaClientJava

EPICS base C++

- https://github.com/epics-base/pvAccessCPP
- https://github.com/epics-base/pvaClientCPP

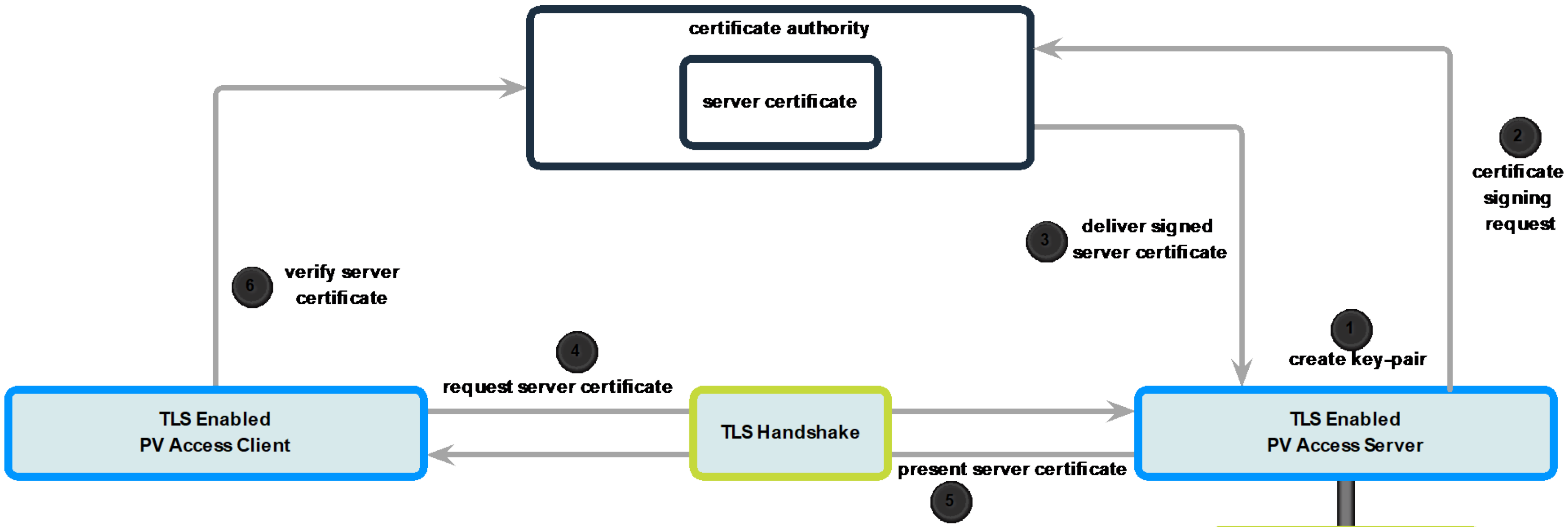
Server Authentication

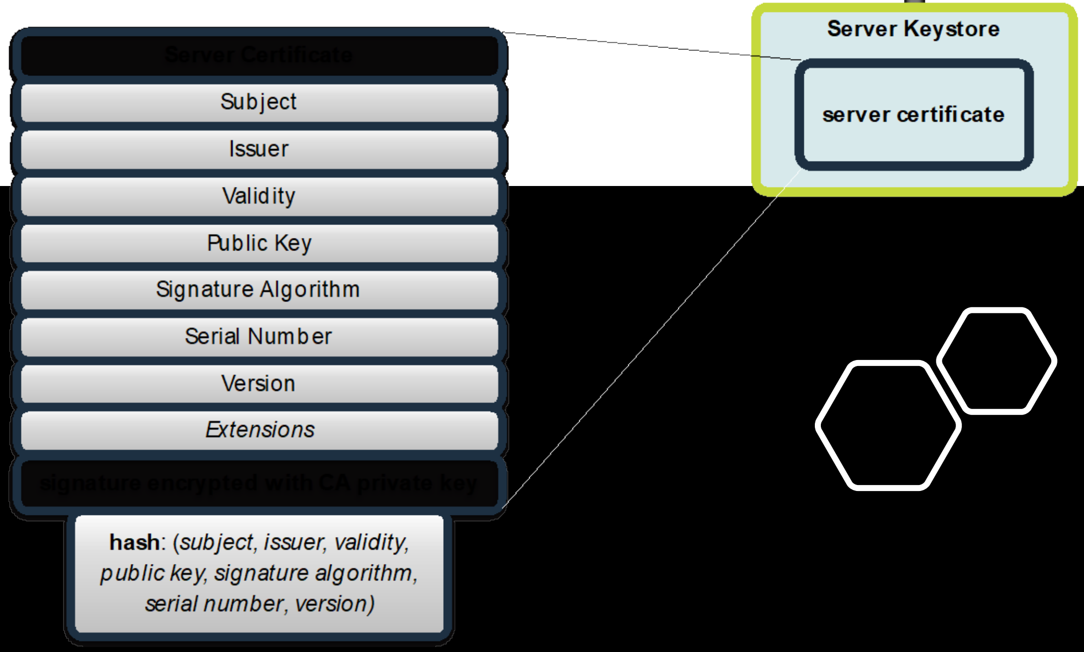

certificate authority
server certificate
2
certificate signing request
3
deliver signed server certificate
6
verify server certificate
1
create key-pair
4
request server certificate
TLS Enabled PV Access Client
TLS Handshake
TLS Enabled PV Access Server
present server certificate
5

Obtain and use Server Certificate

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048
```

```
openssl req -new -key private_key.pem -out server_csr.pem
```


certificate authority
server certificate
2
certificate signing request
3
deliver signed server certificate
6
verify server certificate
1
create key-pair
4
request server certificate
TLS Enabled PV Access Client
TLS Handshake
TLS Enabled PV Access Server
present server certificate
5

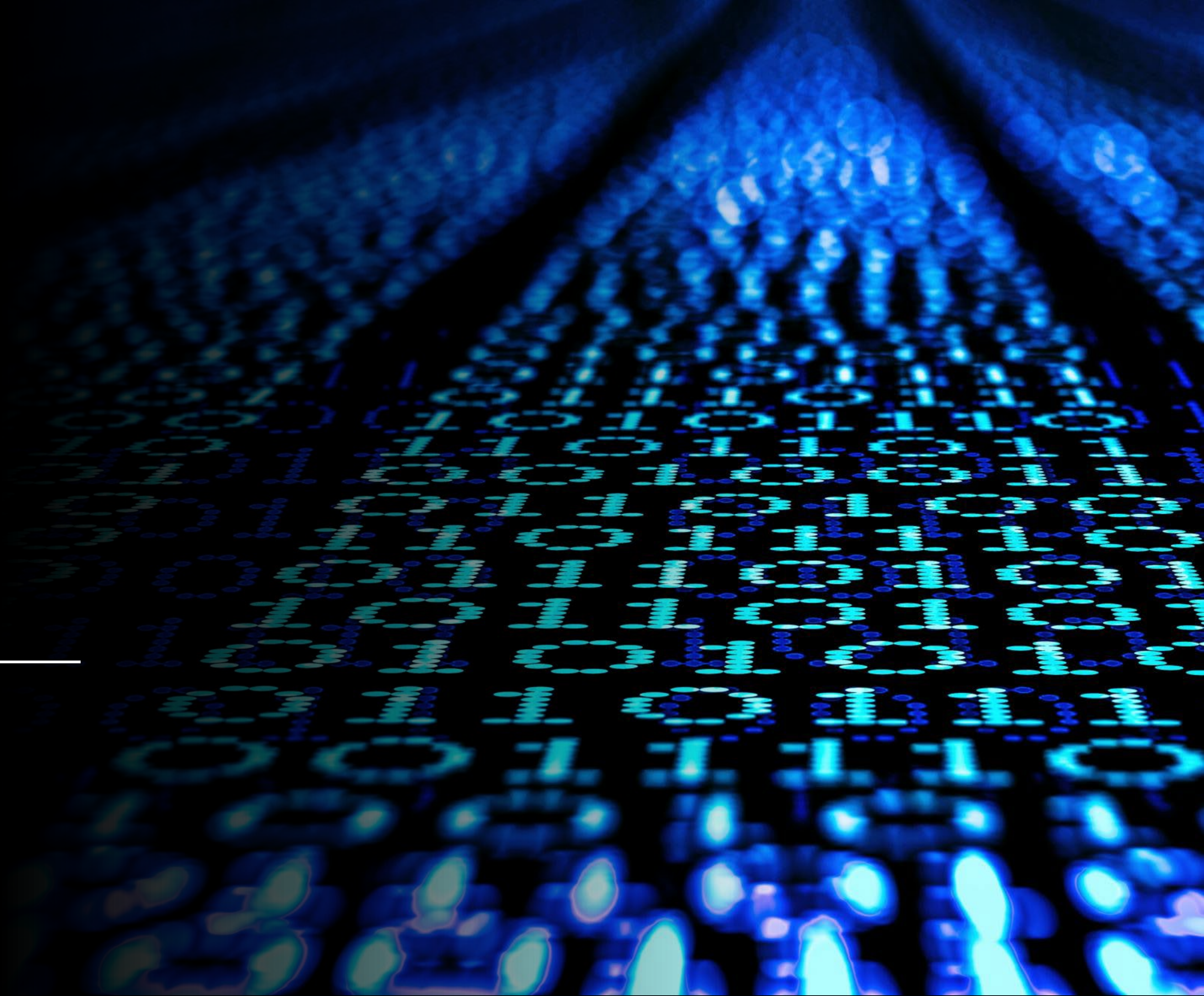


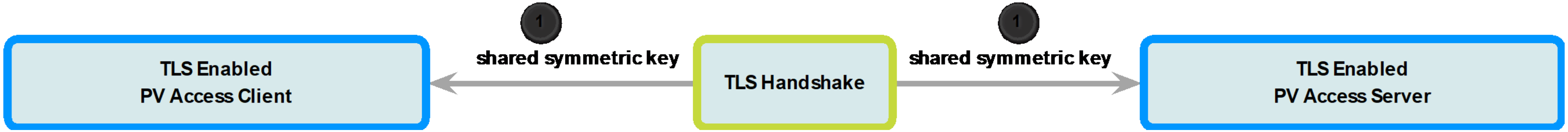
Obtain and use Server Certificate

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt rsa_keygen_bits:2048

openssl req -new -key private_key.pem -out server_csr.pem
```

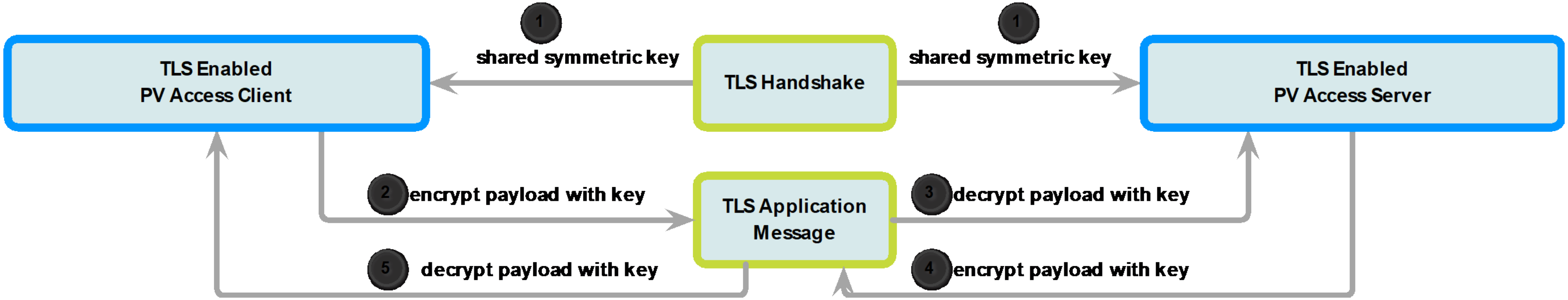
Data Encryption





Data Encryption

- Symmetric Key is generated and securely shared during TLS Handshake



Data Encryption

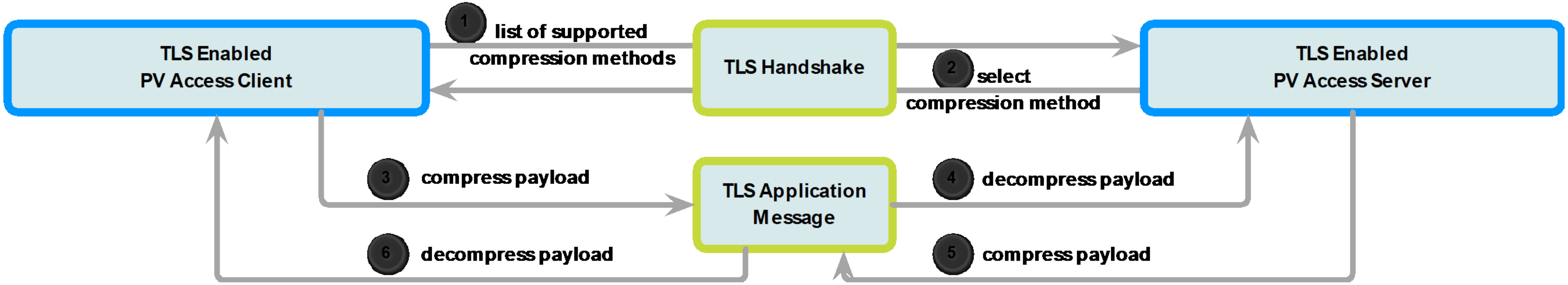
- Symmetric Key is generated and securely shared during **TLS Handshake**
- When PV Access messages are sent inside TLS Application messages, they are encrypted using the key
- The message receiver decrypts the message with the same key
- To an outside observer the packets are protected

Server Certificate Rotation

Server Certificate Rotation

- Server Certificates can have a validity (expiration date)
- In PV Access, it is typical for server connections to be very long lived – sometimes in the order of months and years
- The implementation of the TLS layer will allow new Server Certificates to be rotated in **while connections remain live**
- This is achieved by allowing the Server to initiate a new TLS Handshake by issuing a TLS Hello Message.
- When a Server Certificate (or other parameter of the TLS session) changes we say the TLS Record State has changed

Data Compression



Data Compression

- DEFLATE
- LZ77
- LZW
- LZ4
- zstd
- Brotli
- Snappy

- Compression method agreed during TLS Handshake
- When PV Access messages are sent inside TLS Application messages, they are compressed using the selected method
- The message receiver decompresses the message with the agreed method
- This compression applies to all messages

Client Authentication

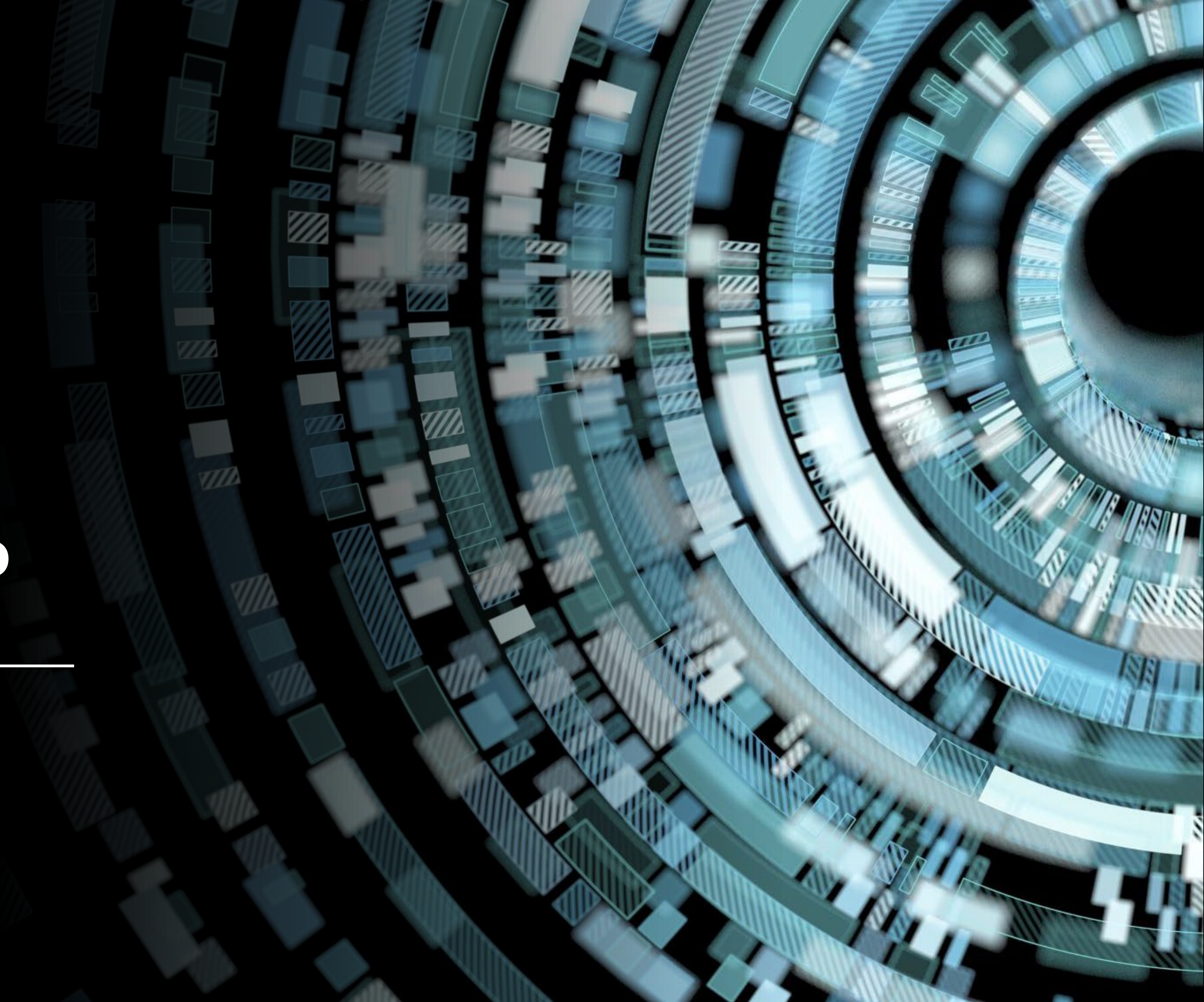
Client Authentication using Client Certificates

- Using Client Certificates, we can verify the identity of clients
- The Certificate contains the public key of the Client, signed by a Certificate Authority
- Certifying Authority is a trusted entity that signs the information contained in the certificate
- The Client Certificate contains a Client's public key
- If configured to do so a Server can request that a Client identify itself during the TLS handshake
- A Client can prove its identity to the Server by signing messages with its private key

Client Authorization

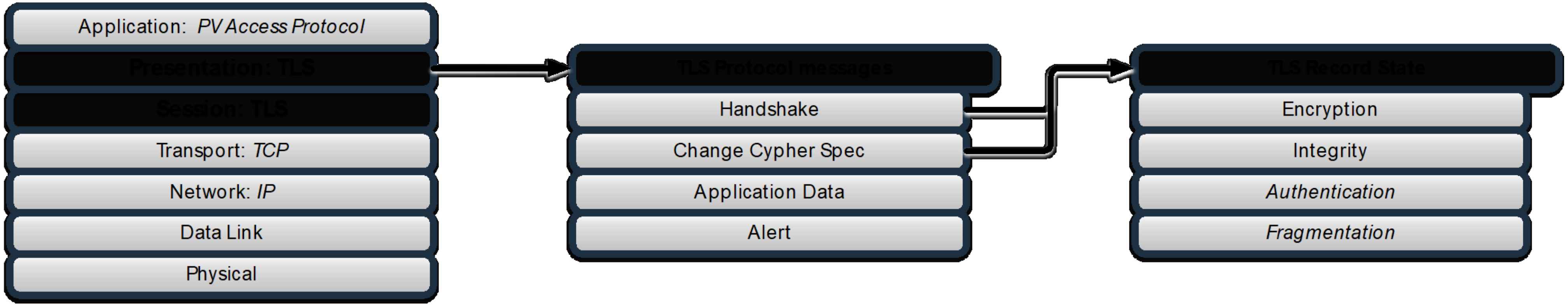
Authorizing Access to PVs and Servers

- Once the identity of a Client is determined from the Client Certificate it can be used to implement strong access control
  - Security configuration for PV Access Server
    - Could configure a white-list of allowed clients for a given PV Access Server or set of PVs
  - Fine Grained PV Access Control - ACF
    - The identify could be used to improve security for EPICS IOCs via ASG (Access Security Group) configuration.
    - Could control access with granularity of request type (get/put/monitor, etc.)
  - Finally, we can imagine connecting to a Directory Service such as Active directory, or LDAP for a centralized approach to a site's security (also using Kerberos)



How it works?

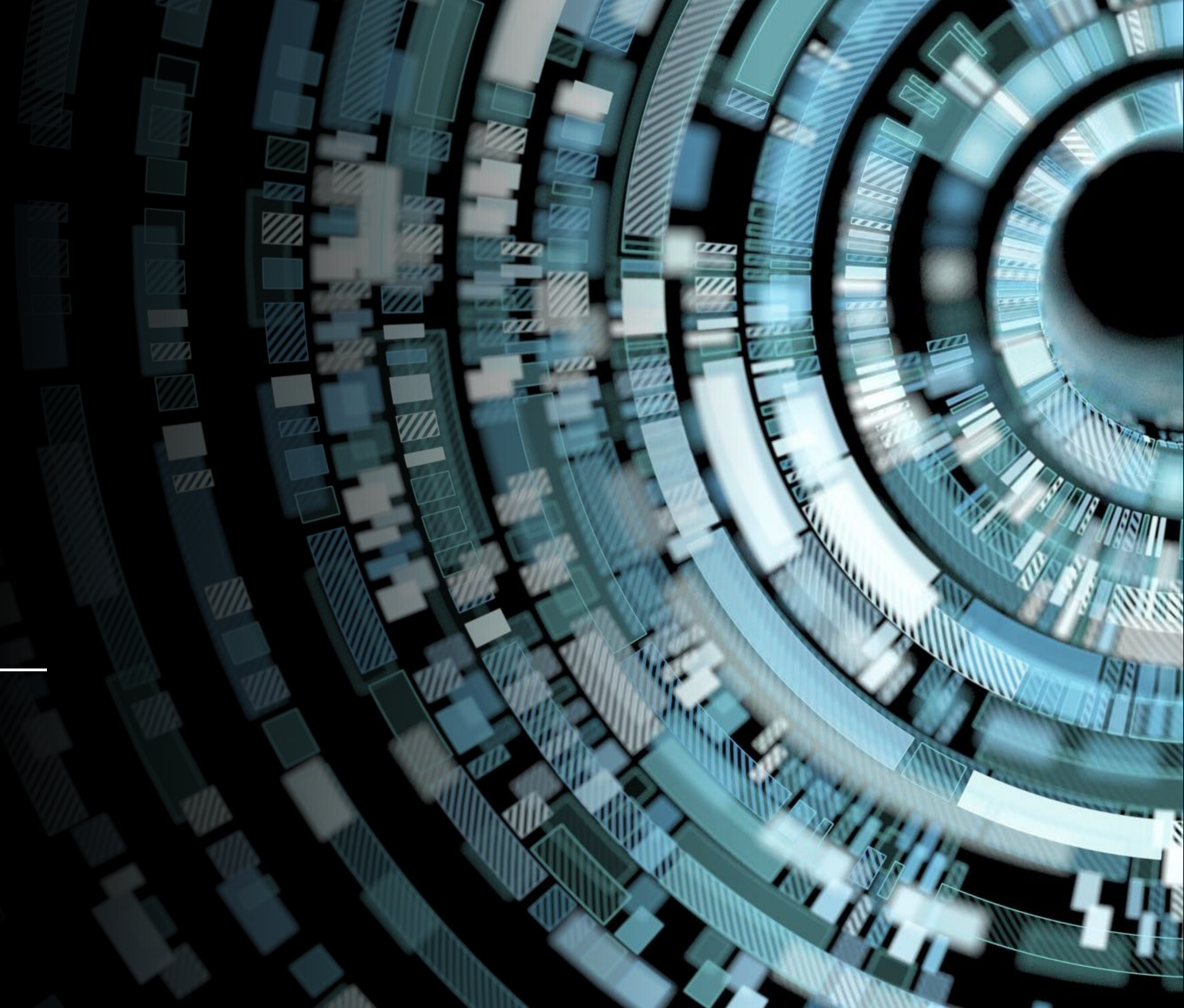
Where does it fit into the communications stack?



TLS in the PV Access Protocol Stack

- ISO OSI Model

- TLS (Transport Layer Security) is encapsulated above the **TCP** communications layer
- **Handshake** and **Change Cypher Spec** messages can update the Record State
- **Application Data** messages contain PV Access messages
- **Alert** messages signal TLS errors
- TLS Record State indicates the selected **Encryption Scheme** and **Certificates** as well as the **Cipher Suite**

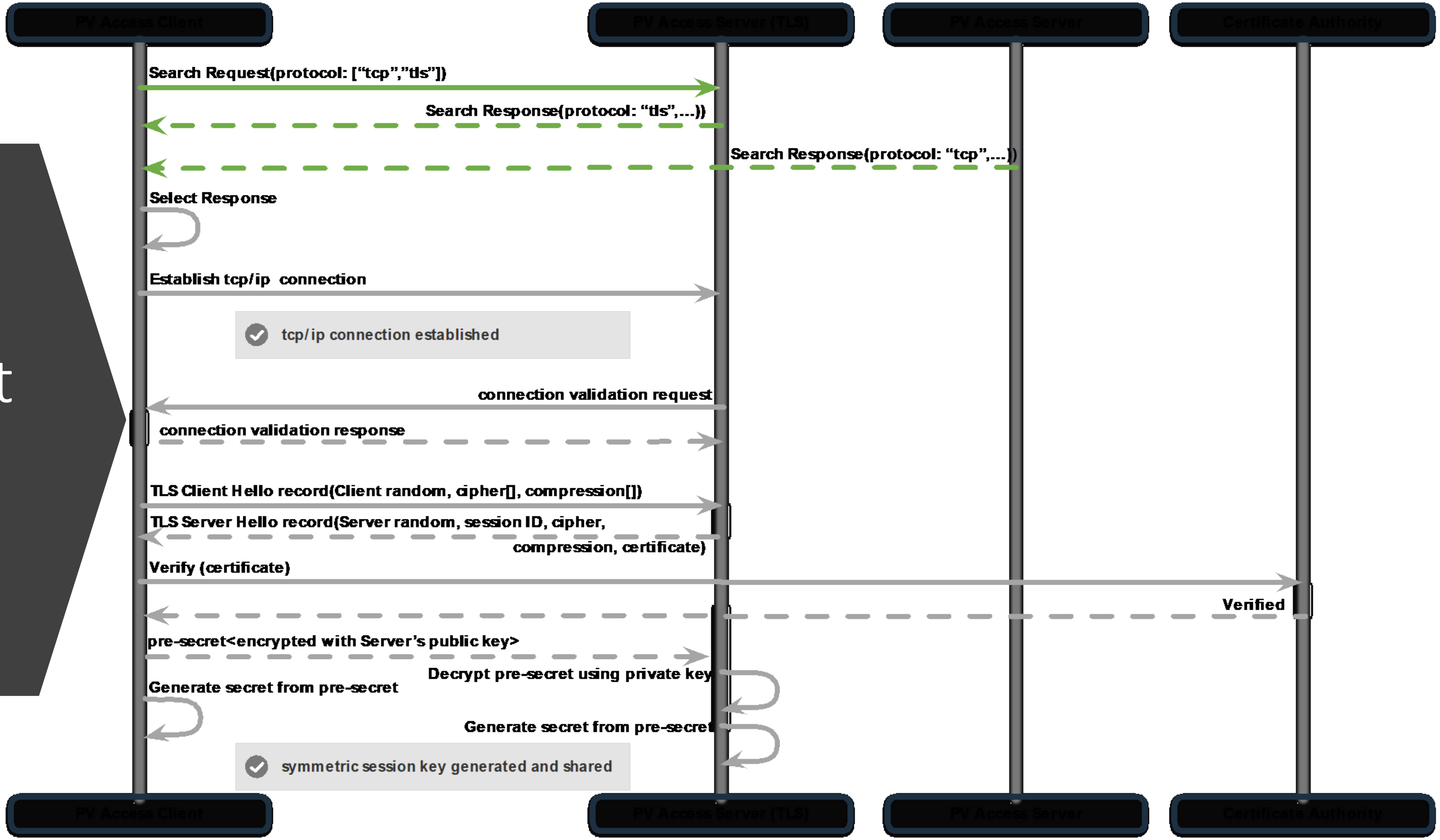


How it works?

Establishing a TLS Session

PV Access Client Connect Sequence Diagram

1. Channel Search



PV Access Client

PV Access Server (TLS)

PV Access Server

Certificate Authority

Search Request(protocol: [“tcp”,”tls”])

Search Response(protocol: “tls”,...))

Search Response(protocol: “tcp”,...))

Select Response

Establish tcp/ip connection

✓ tcp/ip connection established

connection validation request

connection validation response

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

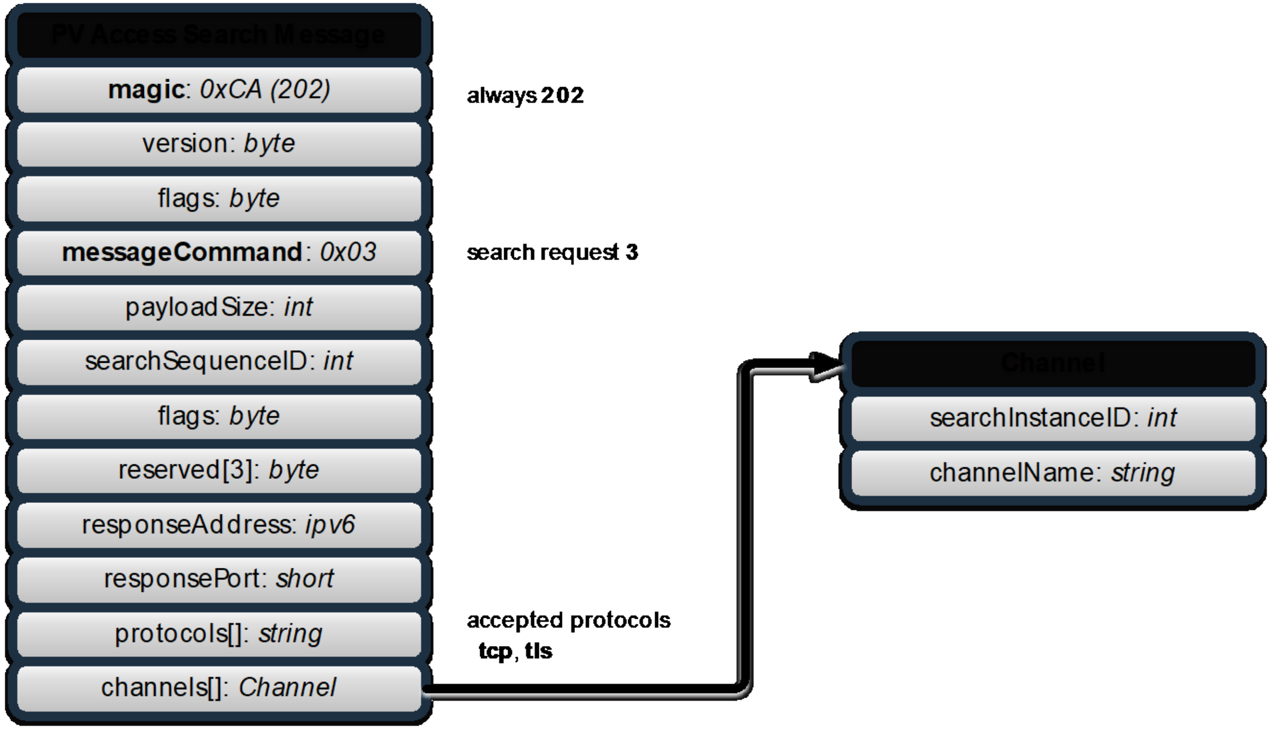
pre-secret<encrypted with Server’s public key>

Decrypt pre-secret using private key

Generate secret from pre-secret

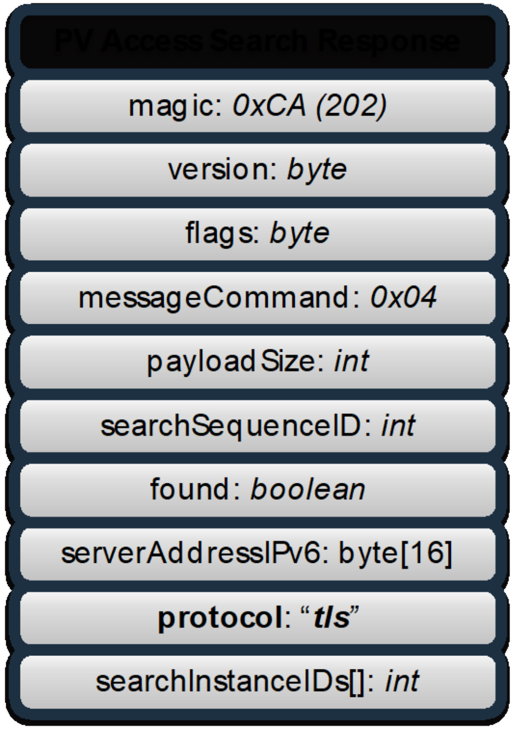
Generate secret from pre-secret

✓ symmetric session key generated and shared



Search requests

- Client requests will now allow **tls** in the protocols array.
- Servers that don’t support TLS features will ignore search requests with exclusively “tls” prototype strings.
- Servers that don’t support TLS features will continue in TCP mode.



PV Access Search Response

magic: *0xCA (202)* — always **202**

version: *byte*

flags: *byte*

messageCommand: *0x04* — search response **4**

payloadSize: *int*

searchSequenceID: *int*

found: *boolean*

serverAddressIPv6: byte[16]

protocol: "***tls***" — I support TLS

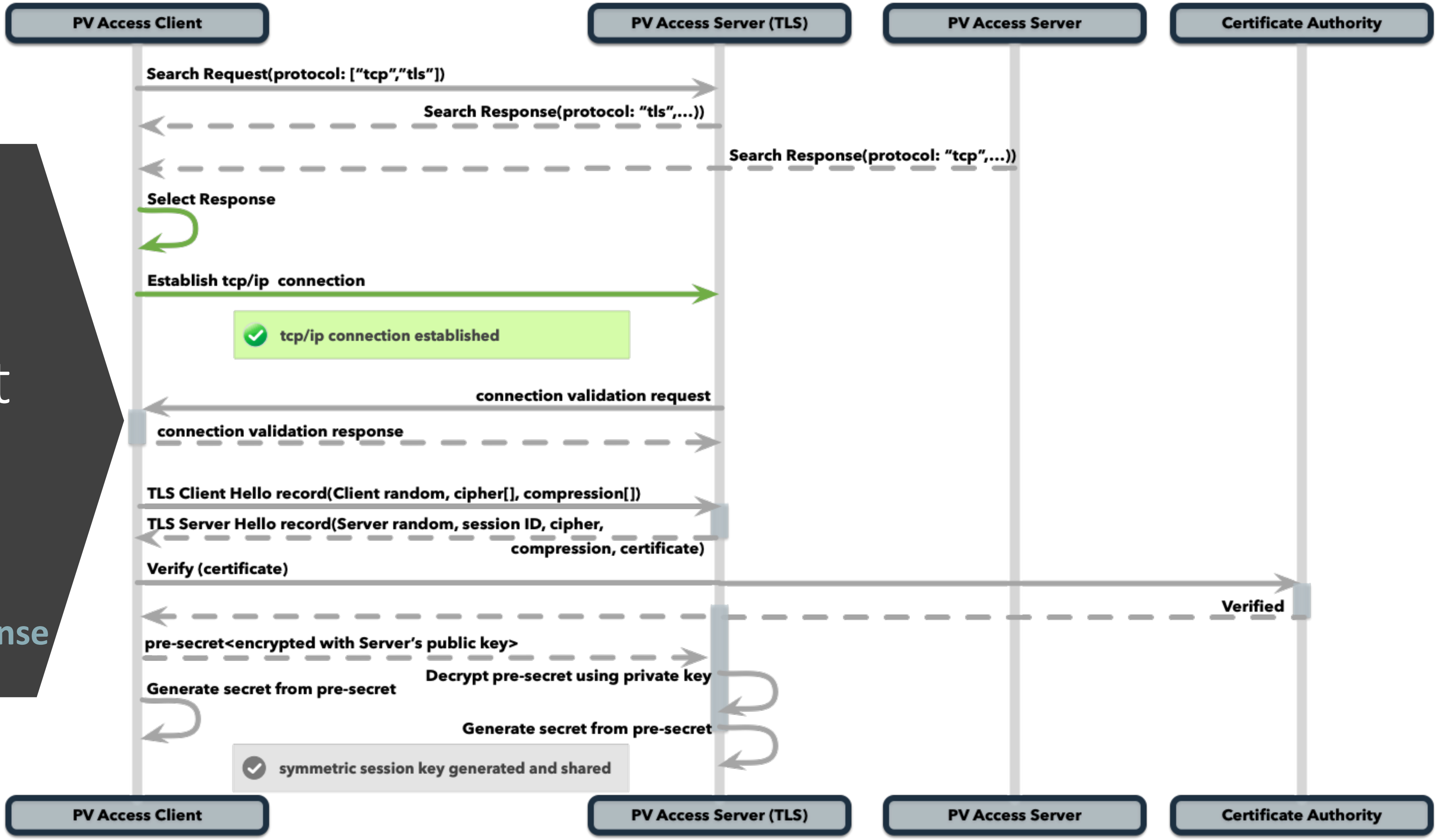
searchInstanceIDs[]: *int*

Search responses

- Servers supporting TLS will return **tls** as the protocol string.
- Clients that don't support TLS will proceed without TLS handshake.

PV Access Client Connect Sequence Diagram

2. Select Protocol Response



PV Access Client

PV Access Server (TLS)

PV Access Server

Certificate Authority

Search Request(protocol: [“tcp”,”tls”])

Search Response(protocol: “tls”,...))

Search Response(protocol: “tcp”,...))

Select Response

Establish tcp/ip connection

✓ tcp/ip connection established

connection validation request

connection validation response

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server’s public key>

Decrypt pre-secret using private key

Generate secret from pre-secret

Generate secret from pre-secret

✓ symmetric session key generated and shared

PV Access Client

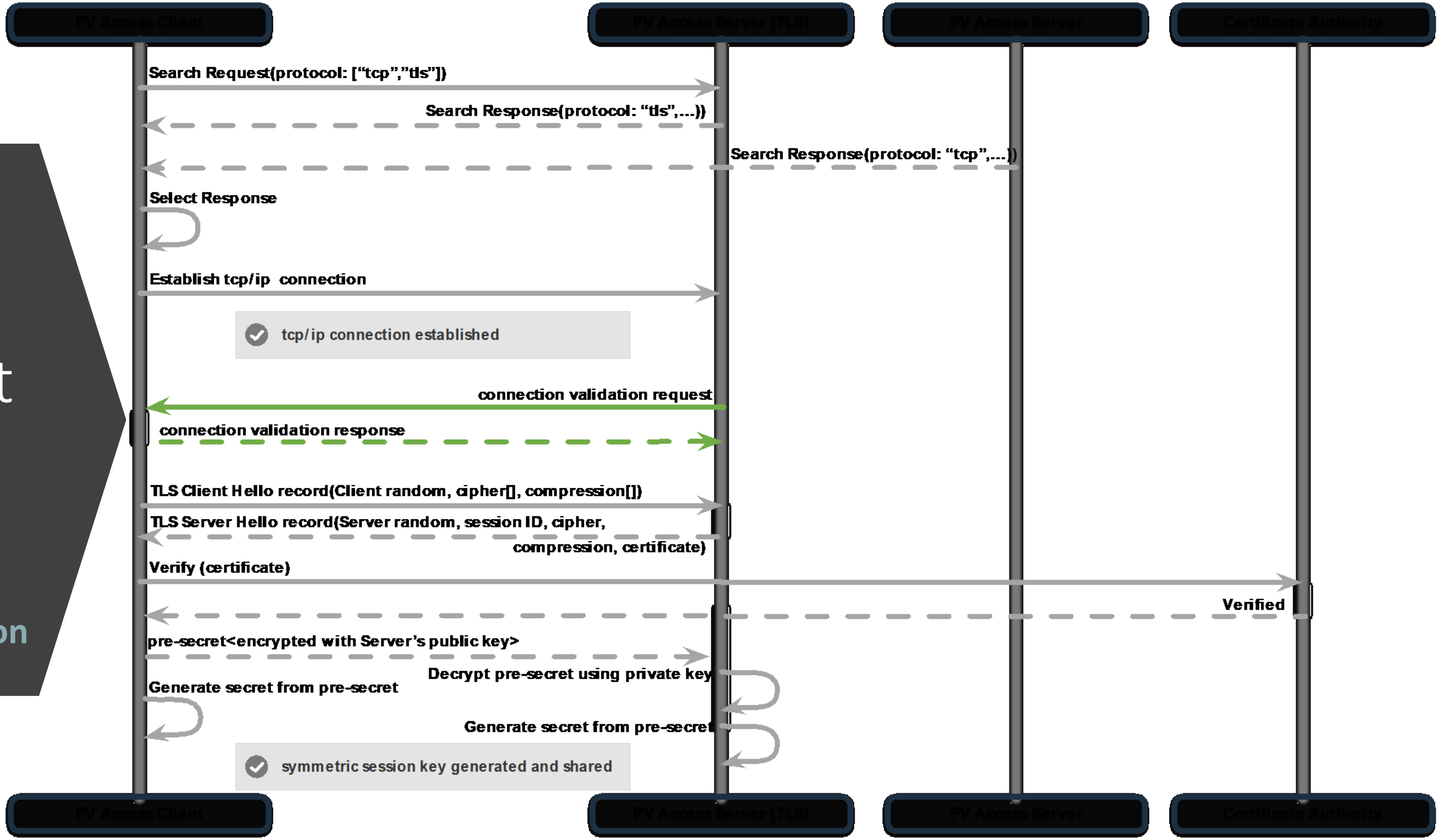
PV Access Server (TLS)

PV Access Server

Certificate Authority

PV Access Client Connect Sequence Diagram

3. Connection Validation



Search Request(protocol: [“tcp”,”tls”])

Search Response(protocol: “tls”,...))

Search Response(protocol: “tcp”,...))

Select Response

Establish tcp/ip connection

✓ tcp/ip connection established

connection validation request

connection validation response

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server’s public key>

Decrypt pre-secret using private key

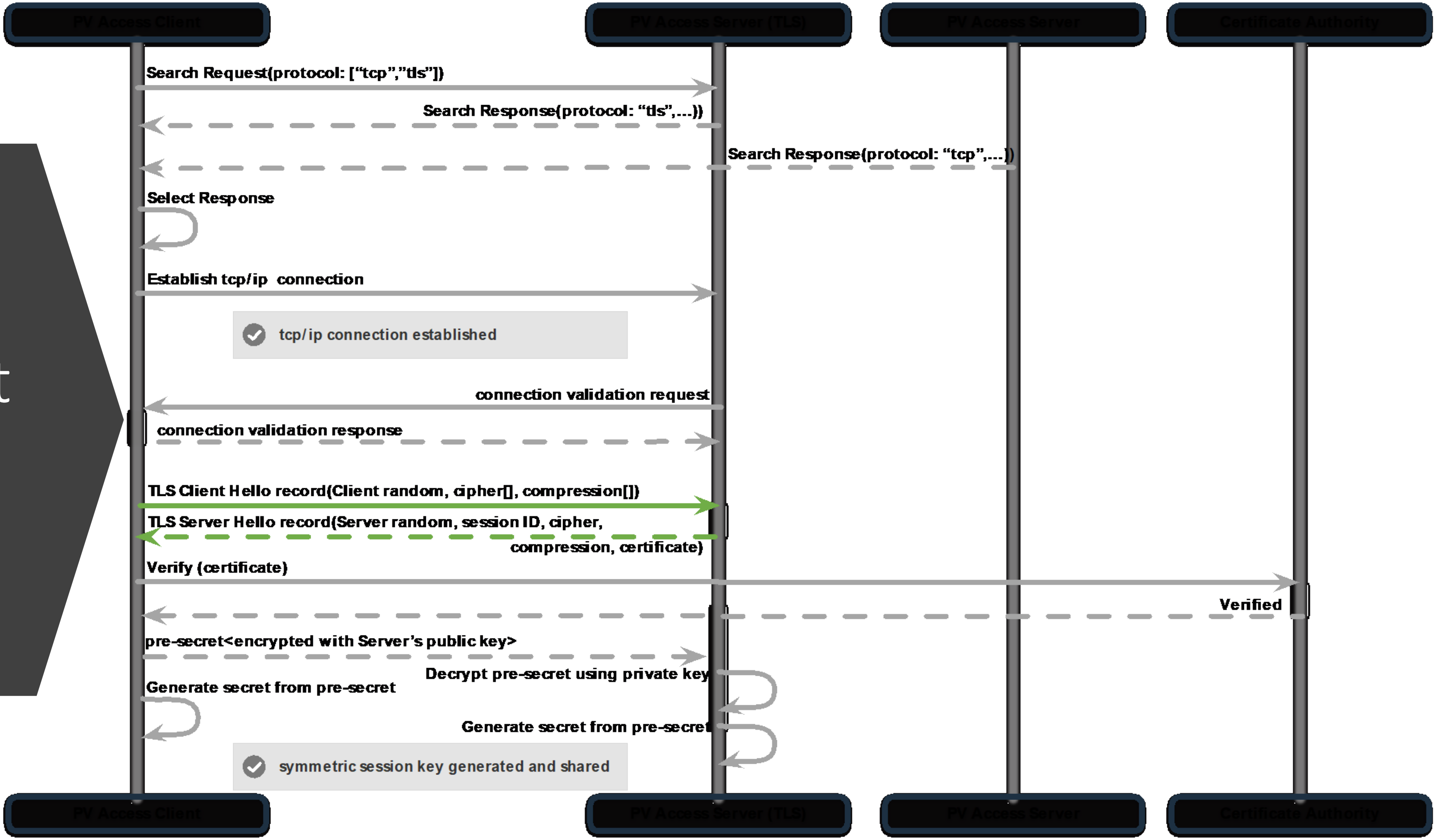
Generate secret from pre-secret

Generate secret from pre-secret

✓ symmetric session key generated and shared

PV Access Client Connect Sequence Diagram

4. TLS Handshake & Certificate



PV Access Client

PV Access Server (TLS)

PV Access Server

Certificate Authority

Search Request(protocol: [“tcp”,”tls”])

Search Response(protocol: “tls”,...))

Search Response(protocol: “tcp”,...))

Select Response

Establish tcp/ip connection

✓ tcp/ip connection established

connection validation request

connection validation response

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server’s public key>

Decrypt pre-secret using private key

Generate secret from pre-secret

Generate secret from pre-secret

✓ symmetric session key generated and shared

PV Access Client

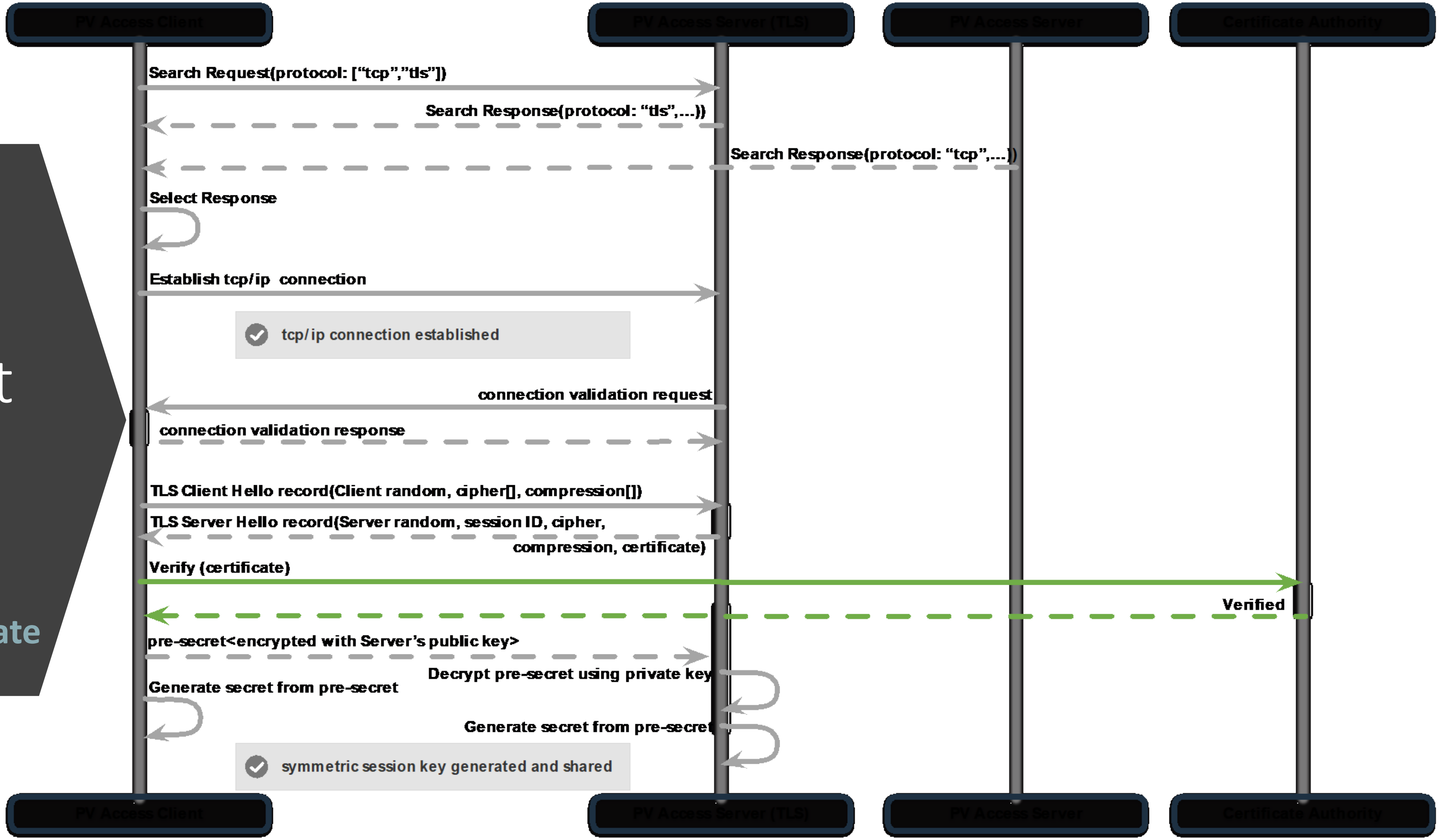
PV Access Server (TLS)

PV Access Server

Certificate Authority

PV Access Client Connect Sequence Diagram

5. Verify Server Certificate



Search Request(protocol: [“tcp”,”tls”])

Search Response(protocol: “tls”,...))

Search Response(protocol: “tcp”,...))

Select Response

Establish tcp/ip connection

tcp/ip connection established

connection validation request

connection validation response

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server’s public key>

Decrypt pre-secret using private key

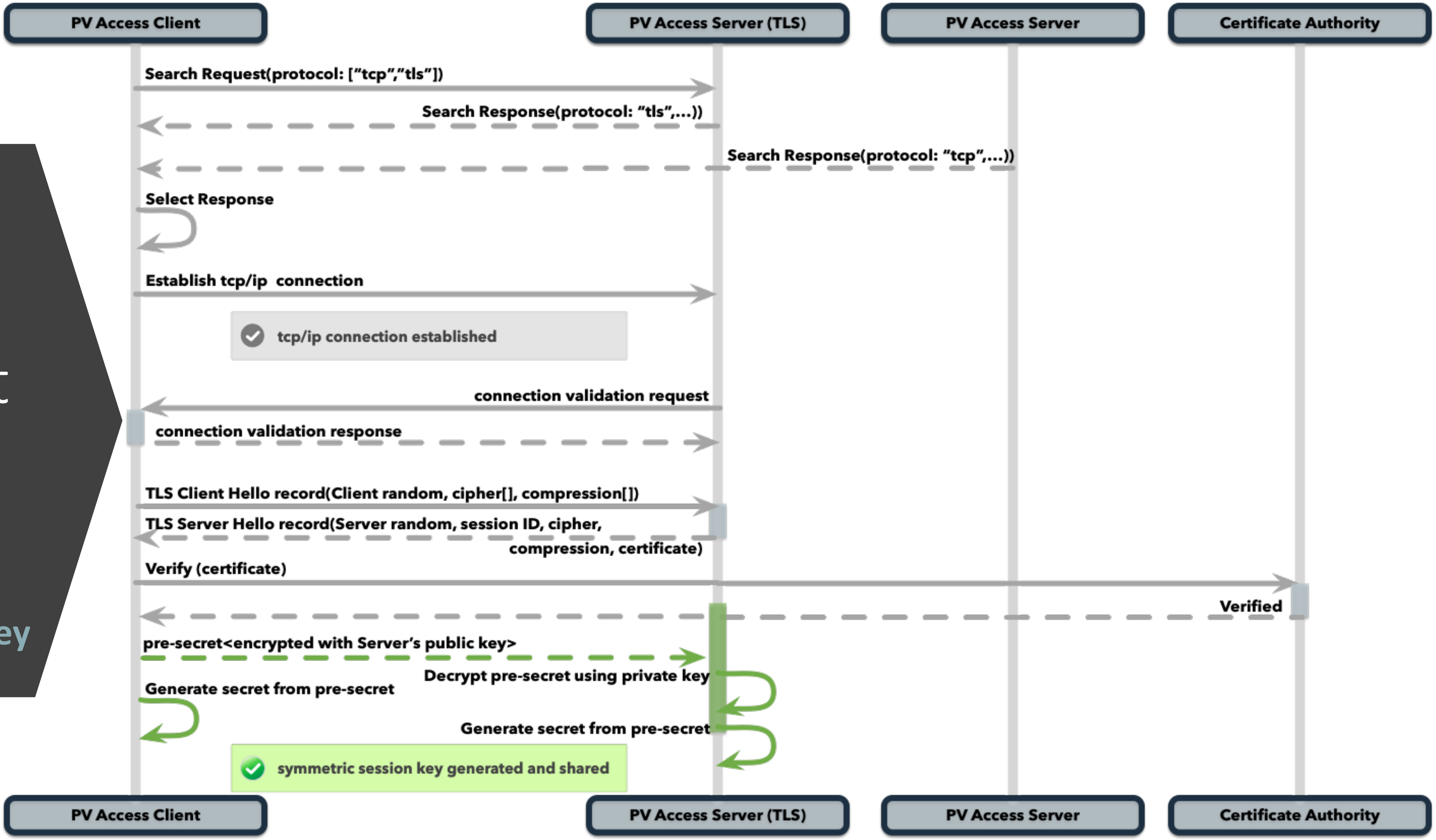
Generate secret from pre-secret

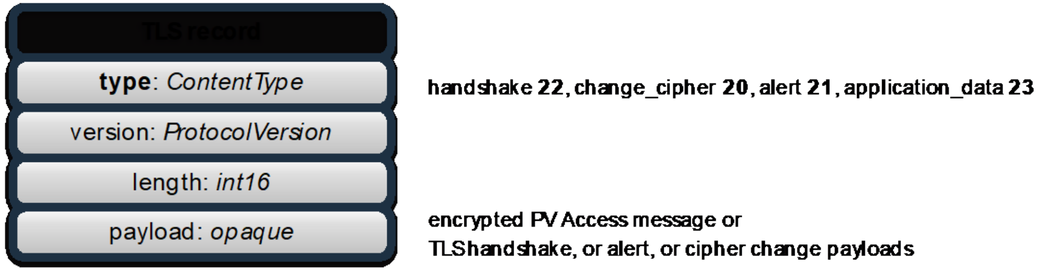
Generate secret from pre-secret

symmetric session key generated and shared

PV Access Client Connect Sequence Diagram

6. Symmetric Session Key


PV Access Client
PV Access Server (TLS)
PV Access Server
Certificate Authority
Search Request(protocol: ["tcp","tls"])
Search Response(protocol: "tls",...))
Search Response(protocol: "tcp",...))
Select Response
Establish tcp/ip connection
tcp/ip connection established
connection validation request
connection validation response
TLS Client Hello record(Client random, cipher[], compression[])
TLS Server Hello record(Server random, session ID, cipher, compression, certificate)
Verify (certificate)
Verified
pre-secret<encrypted with Server's public key>
Decrypt pre-secret using private key
Generate secret from pre-secret
Generate secret from pre-secret
symmetric session key generated and shared
PV Access Client
PV Access Server (TLS)
PV Access Server
Certificate Authority



TLS record

type: *ContentType*

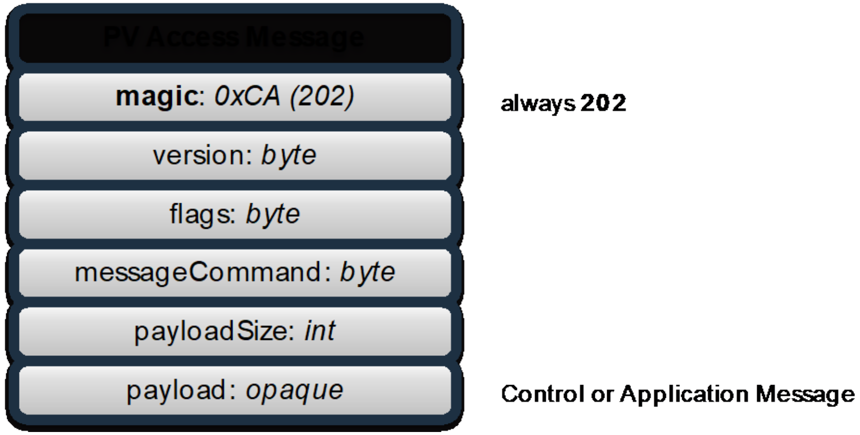
version: *ProtocolVersion*

length: *int16*

payload: *opaque*

handshake **22**, change_cipher **20**, alert **21**, application_data **23**

encrypted PV Access message or
TLS handshake, or alert, or cipher change payloads



PV Access Message

magic: *0xCA (202)*

version: *byte*

flags: *byte*

messageCommand: *byte*

payloadSize: *int*

payload: *opaque*

always **202**

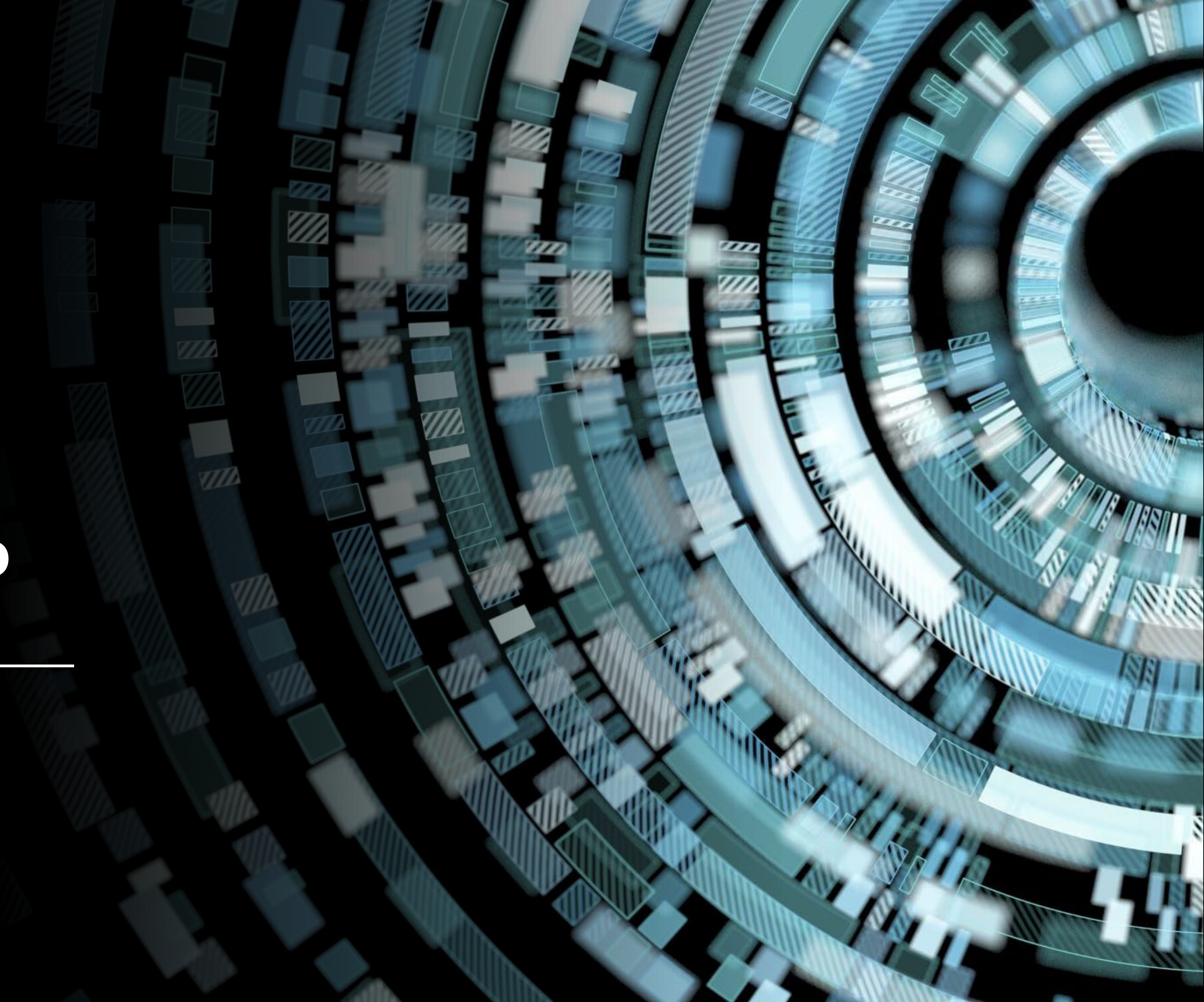
Control or Application Message

TLS session

- TLS record and the PV Access message both start with a single magic byte
- If it is **202** then it is a regular PV Access message.
- If it is one of the valid TLS message types (**22, 20, 21 or 23**), then it is a TLS message.
- PV Access Servers that don’t support TLS will error out when receiving an invalid magic code

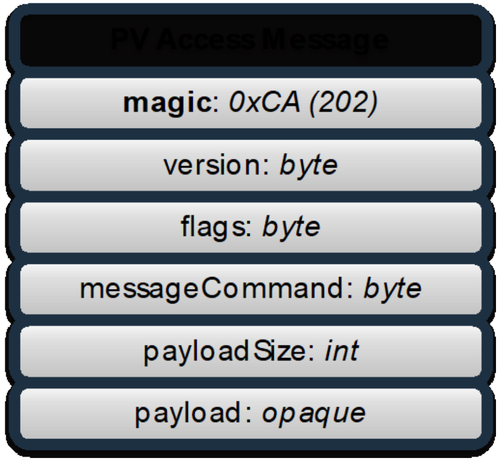
How it works?

Encapsulating PV Access Messages



TLS Encapsulation

Clients		Servers
- Encapsulate PV Access application and control messages inside TLS records	→	- Unwrap TLS records and then process the PV Access messages in the normal way.
- Unwrap TLS records and then process the PV Access messages in the normal way.	←	- Encapsulate PV Access response messages inside TLS records.



PV Access Message

magic: *0xCA (202)*

version: *byte*

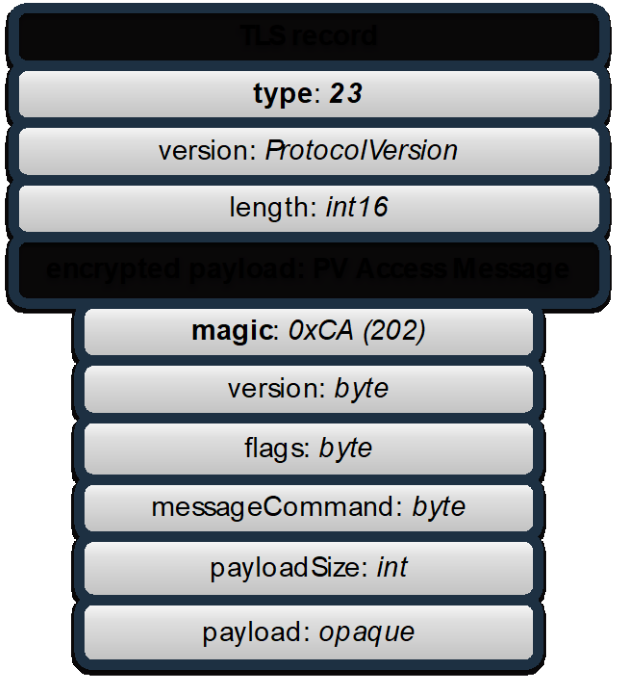
flags: *byte*

messageCommand: *byte*

payloadSize: *int*

payload: *opaque*

Without TLS Support
Send Bare PV Access Message



TLS record

type: *23*

version: *ProtocolVersion*

length: *int16*

encrypted payload: PV Access Message

magic: *0xCA (202)*

version: *byte*

flags: *byte*

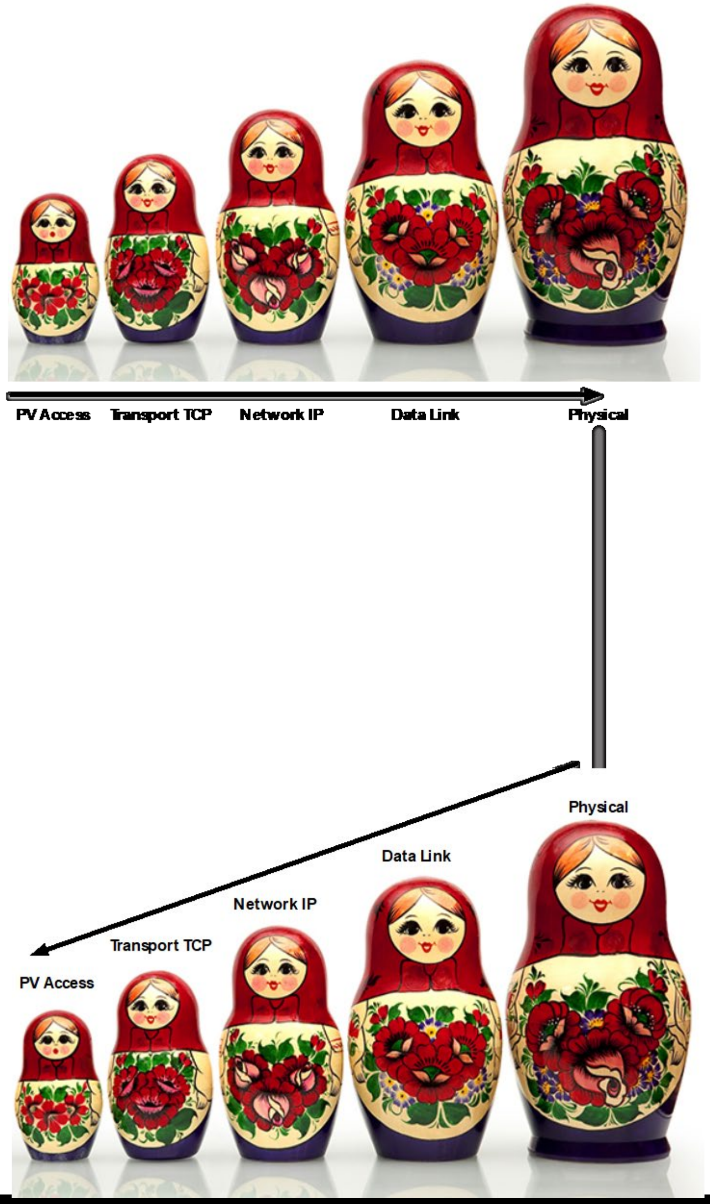
messageCommand: *byte*

payloadSize: *int*

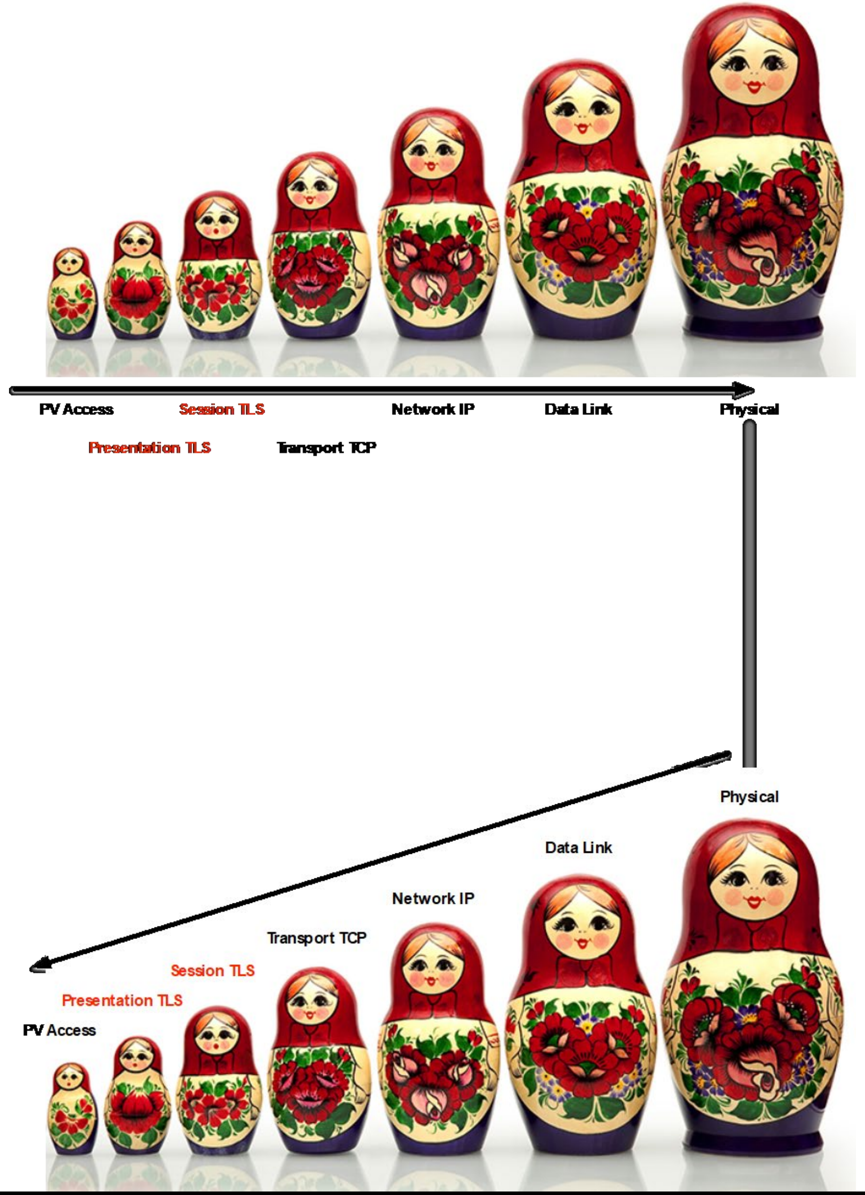
payload: *opaque*

With TLS Support
PV Access Messages
Encapsulated inside TLS record

Encapsulation of PV Access Messages

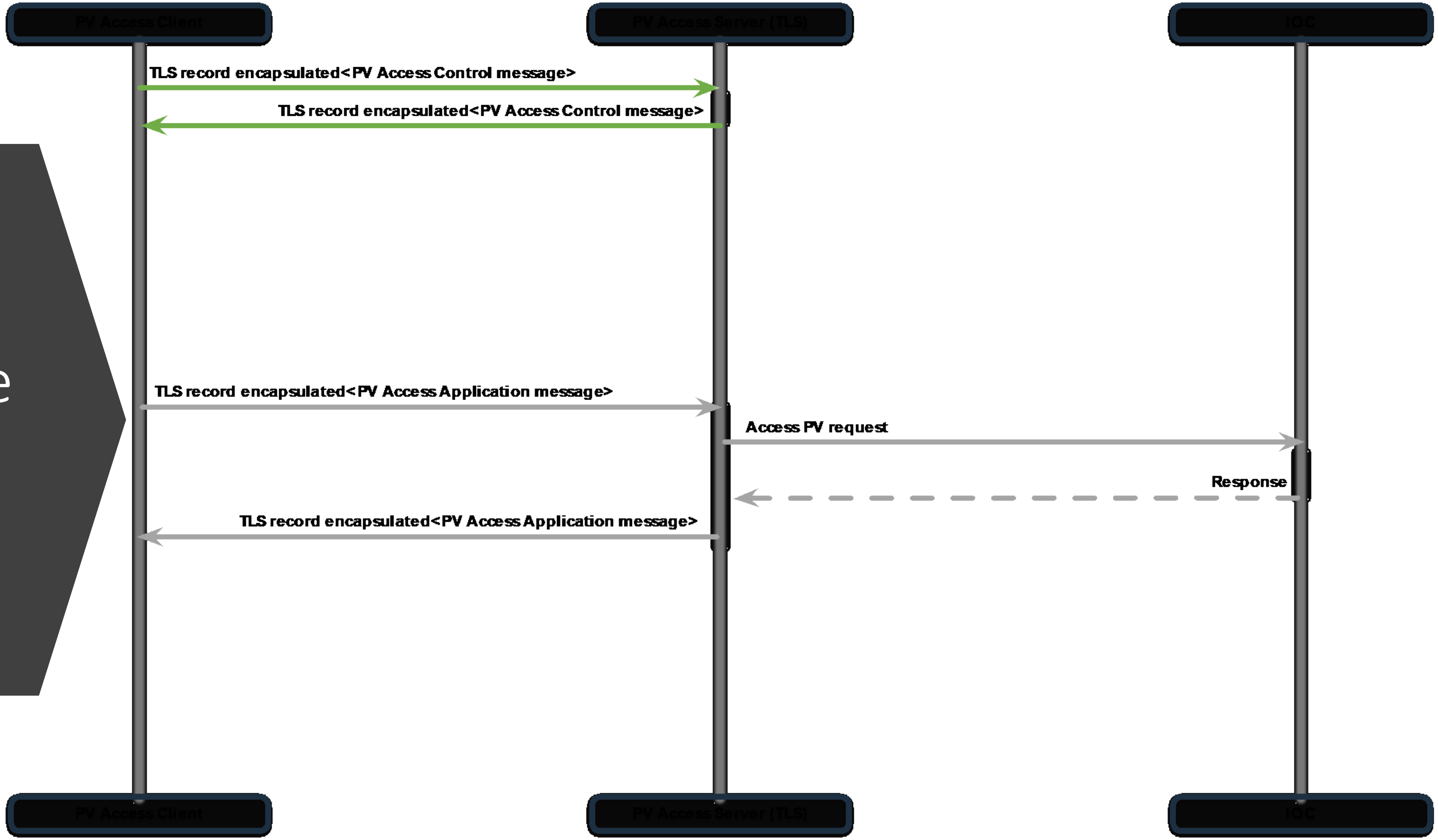


Encapsulation of PV Access Messages + TLS



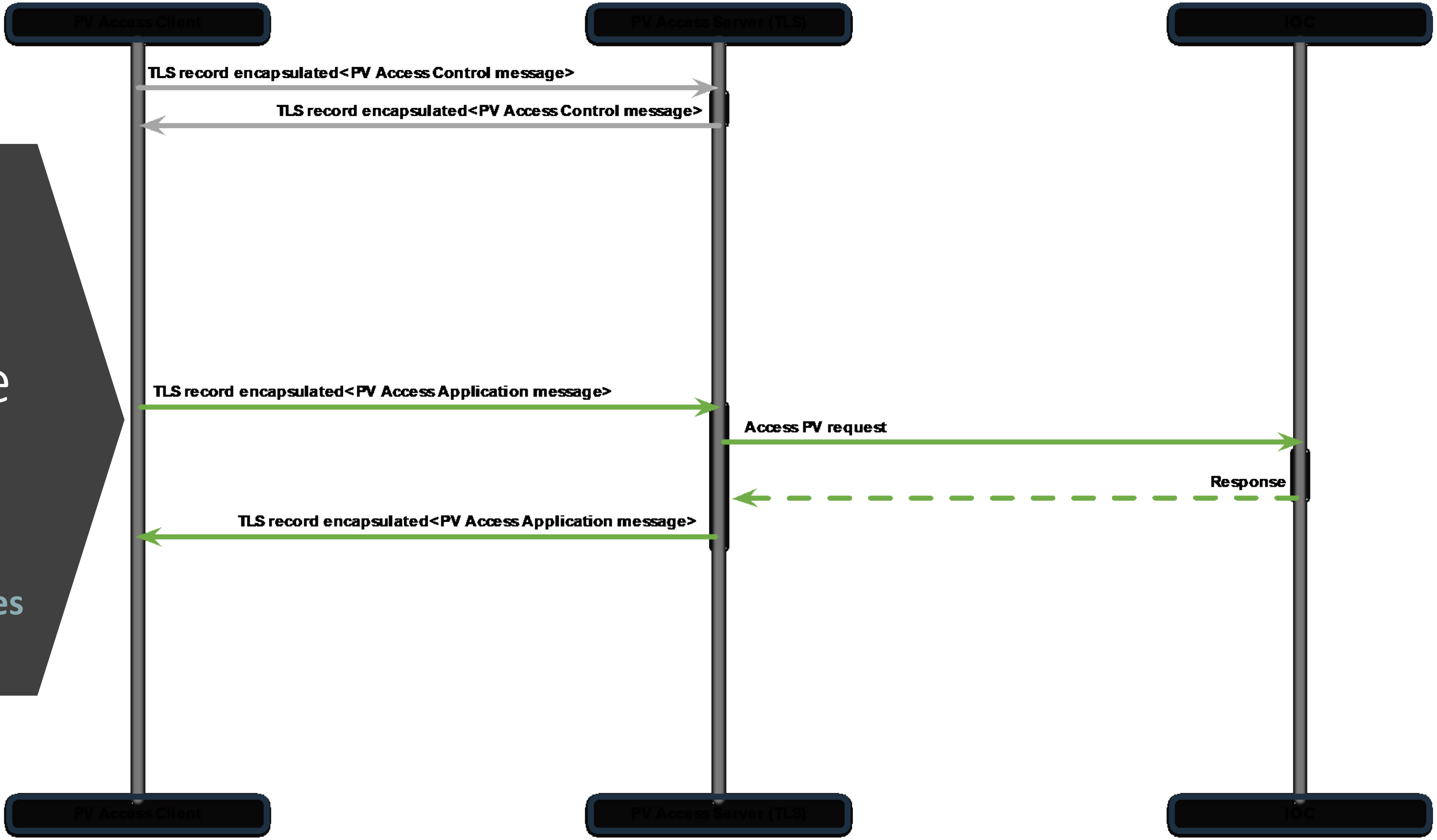
PV Access Data Exchange Sequence Diagram

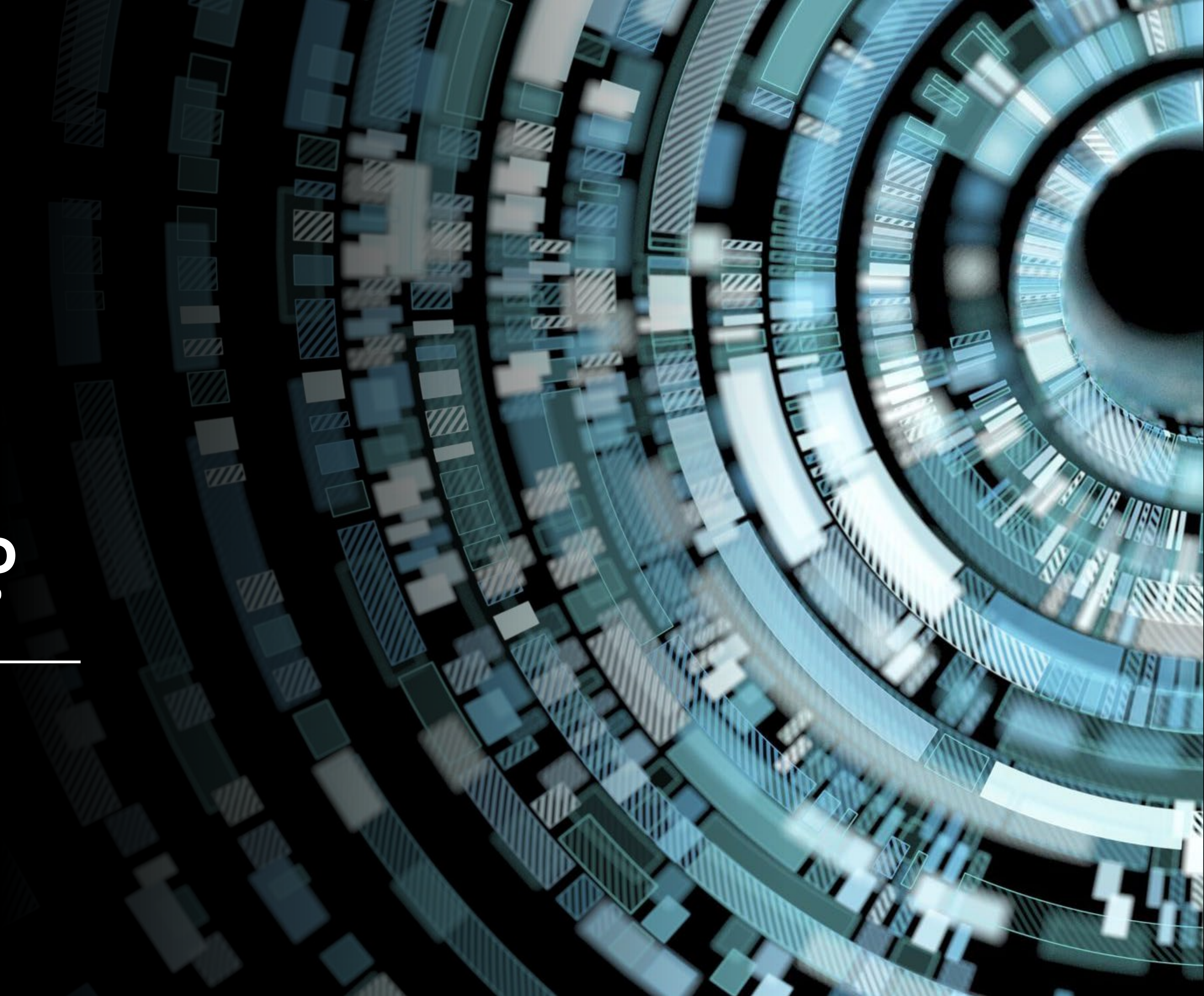
PVA Control Messages


PV Access Client
PV Access Server (TLS)
IOC
TLS record encapsulated<PV Access Control message>
TLS record encapsulated<PV Access Control message>
TLS record encapsulated<PV Access Application message>
Access PV request
Response
TLS record encapsulated<PV Access Application message>
PV Access Client
PV Access Server (TLS)
IOC

PV Access Data Exchange Sequence Diagram

PVA Application Messages


PV Access Client
PV Access Server (TLS)
IOC
TLS record encapsulated<PV Access Control message>
TLS record encapsulated<PV Access Control message>
TLS record encapsulated<PV Access Application message>
Access PV request
Response
TLS record encapsulated<PV Access Application message>
PV Access Client
PV Access Server (TLS)
IOC

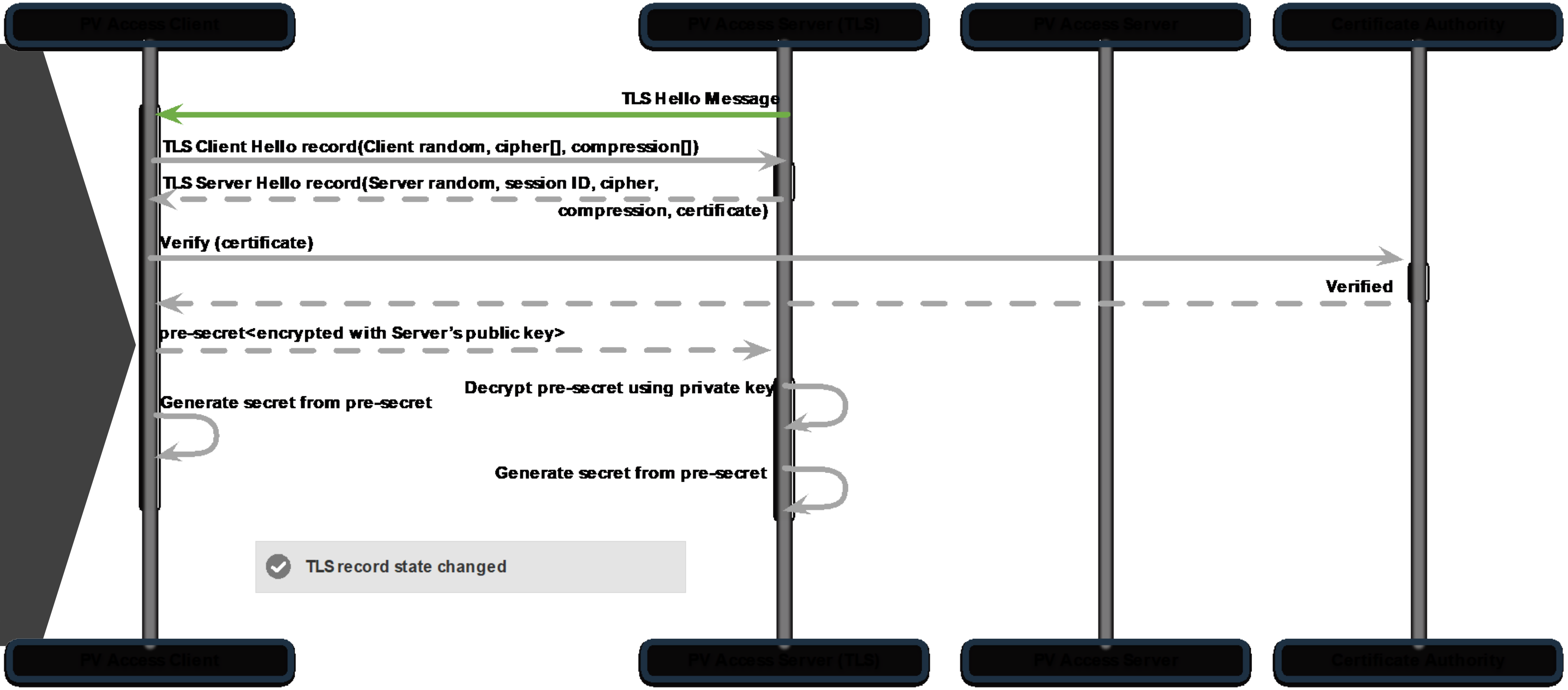


How it works?

Server Certificate Rotation

Server Certificate Exchange Sequence Diagram

1. Initiate with Hello Message from Server



PV Access Client

PV Access Server (TLS)

PV Access Server

Certificate Authority

TLS Hello Message

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server's public key>

Decrypt pre-secret using private key

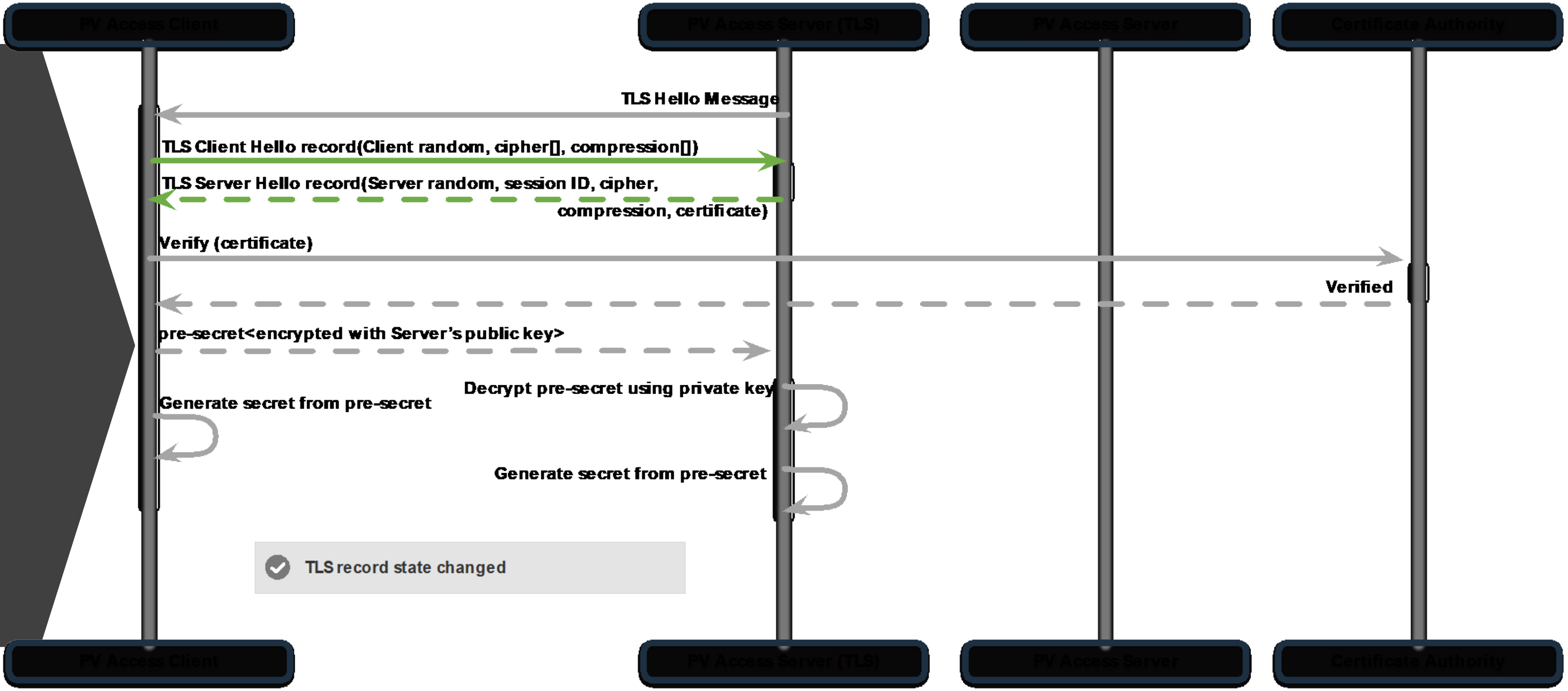
Generate secret from pre-secret

Generate secret from pre-secret

✓ TLS record state changed

Server Certificate Exchange Sequence Diagram

2. TLS Handshake & Certificate



TLS Hello Message

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server's public key>

Decrypt pre-secret using private key

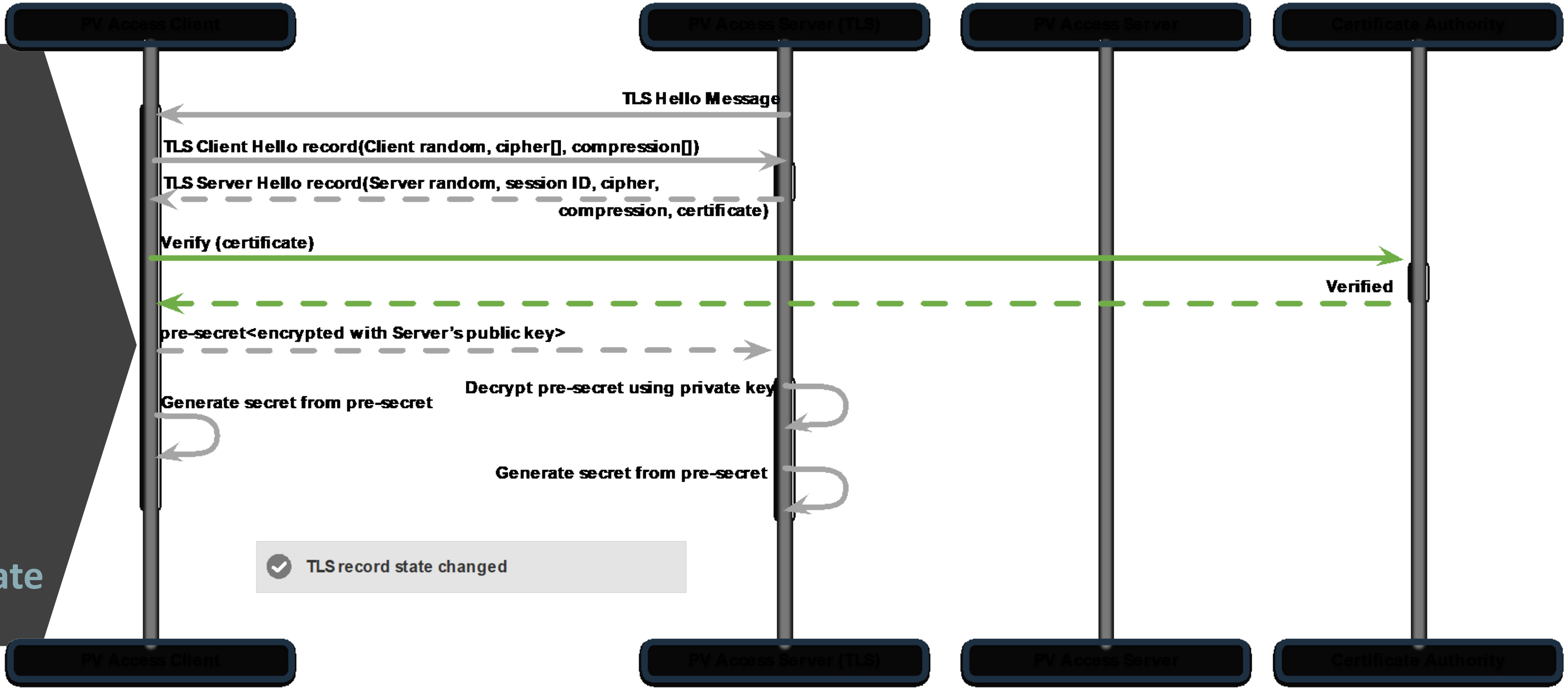
Generate secret from pre-secret

Generate secret from pre-secret

✓ TLS record state changed

Server Certificate Exchange Sequence Diagram

3. Verify Server Certificate



TLS Hello Message

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server's public key>

Decrypt pre-secret using private key

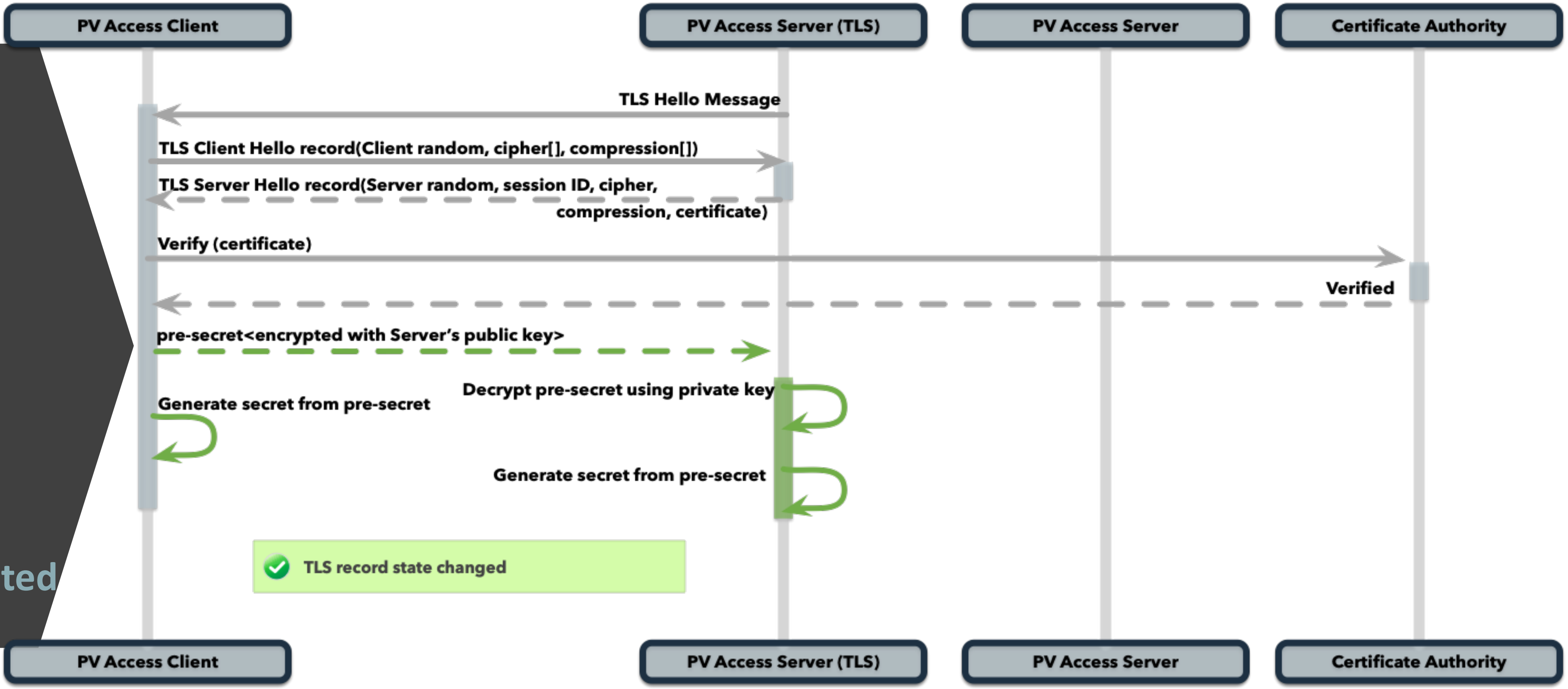
Generate secret from pre-secret

Generate secret from pre-secret

✓ TLS record state changed

Server Certificate Exchange Sequence Diagram

4. Server Certificate Rotated



PV Access Client

PV Access Server (TLS)

PV Access Server

Certificate Authority

TLS Hello Message

TLS Client Hello record(Client random, cipher[], compression[])

TLS Server Hello record(Server random, session ID, cipher, compression, certificate)

Verify (certificate)

Verified

pre-secret<encrypted with Server's public key>

Decrypt pre-secret using private key

Generate secret from pre-secret

Generate secret from pre-secret

✓ TLS record state changed

PV Access Client

PV Access Server (TLS)

PV Access Server

Certificate Authority

PV Access TLS from Osprey DCS

- Thank You

George McIntyre
george@level-n.com

What will adding TLS get us?

TLS Benefits

- Server Certificates →
  - Prevent **Service Impersonation**
  - Prevent **Man-in-the-Middle** attacks
- Cipher suite Message Authentication Codes →
  - Guarantee **Data Integrity**
- Securely shared Symmetric Session Keys →
  - Prevent **Packet Snooping**
- Client Certificates →
  - Provide a mechanism for **Service Access Control**
  - **Protect Data** by allowing Services to Restrict Access
  - Can be used as part of strategy to **Reduce impact of DoS Attacks**

TLS Will Not

- Prevent **PV Impersonation** in a mixed TLS/TCP network
- Prevent discovery of **Service Endpoint** or **PV name**
- Prevent discovery of **Encryption Type**
- Prevent discovery of **Data Transmission Frequency**
- Prevent discovery of approximate **Amount of data transmitted**

TLS in EPICS

Programmatic Interface changes

Possibility of specifying “tls” as a protocol

New PV Access Server configuration options for TLS

Network Management Impact

Install Server Certificates

Configure Network for TLS traffic (network management tools)

***Note**: TLS and legacy clients/servers use the same ports and may interoperate on the same network without any changes to legacy clients and servers*

New EPICS Features

Verified identity of EPIC channels

Guaranteed data integrity for EPICS services/channels

Encrypted EPICs data packets

Unlimited EPICS data size

Data compression supported (not recommended)

Protocol Changes

TLS handshake phase after connection validation

Encapsulation of PV Access Messages

Server Certificate Rotation