HUMBOLDT-
UNIVERSITÄT
ZU BERLIN

FIDO2 TLS 1.3 Extension:
Strong EAP-TLS Authentication for 802.1X Networks

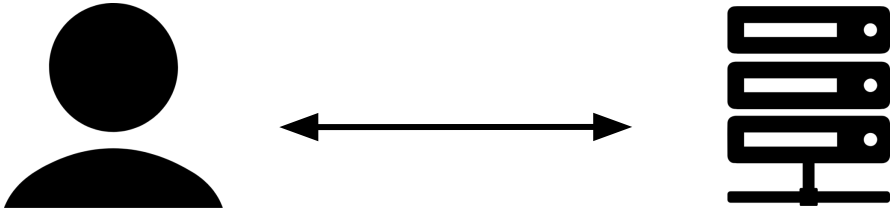
Master's Thesis Defense
Jonas Panizza

Outline

- Motivation
- State of the Art: Client Authentication
  - FIDO2 / WebAuthn
  - TLS 1.3
- FIDO2 as TLS 1.3 Extension
  - Design Principles
  - Integration into TLS Handshake
  - Message Structure & Encoding
  - Control Mechanisms for Key Registration
  - TLS Alerts & Communication to the Application Layer
- Implementation
  - fidoSSL
  - hostap-fido2
- Live Demo with Q&A

Motivation

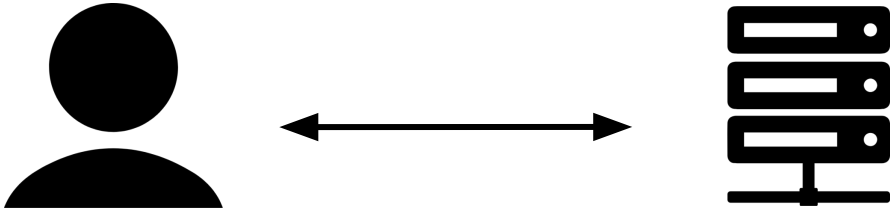
Zero trust on the Internet!



- Reliable authentication of **user** ↔ **service** is (always) needed
- TLS provides: confidentiality, integrity, authenticity
- Server (== service) authentication is common practice in web-environments
  - In TLS implemented with Server-Certificates + PKI
- What about client (==user) authentication?

Motivation

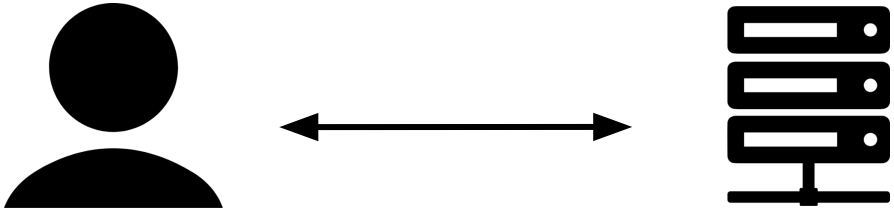
Client Authentication



- No client authentication
  - Static websites, public information, no user-specific interactions, low security applications
- Passwords & OTPs
  - Post TLS handshake authentication, hard to remember, phishing, keylogger, MitM, credential stuffing etc.
- Second Factor
  - Dynamic Authentication, bad usability, often not bound to TLS channel, phishing
- Client Certificates + PKI
  - Barely used in web-environments, hard to manage & deploy
- FIDO2 / WebAuthn
  - Challenge / Response protocol, asymmetric cryptography, secure hardware as keystore, phishing protection, privacy preserving (one public key per service)

Motivation

Client Authentication



- No client authentication
  - Static websites, public information, no user-specific interactions, low security applications
- Passwords & OTPs
  - Post TLS handshake authentication, hard to remember, phishing, keylogger, MitM, credential stuffing etc.
- Second Factor
  - Dynamic Authentication, bad usability, often not bound to TLS channel, phishing
- Client Certificates + PKI
  - Barely used in web-environments, hard to manage & deploy
- FIDO2 / WebAuthn
  - Challenge / Response , post TLS handshake authentication, only for web-environments (and SSH), secure hardware as keystore, phishing protection, privacy preserving (one public key per service)

Motivation

We want fido !! But ..

- FIDO is developed for **HTTPS** (and SSH). What about other TLS-based protocols like ..
  - imaps, pops, smtps
  - ldaps
  - mqtts, nntps, ftps, tftps
  - OpenVPN
  - EAP-TLS
  - Any generic TLS socket
- FIDO does not honor the hierarchical approach of the OSI model
  - TLS responsible for confidentiality, integrity at OSI layer 3
  - Post TLS handshake FIDO authentication at OSI layer 4 → **Cross layer design**
  - No conceptual separation between the functions of each OSI layer
- Why must each application protocol implement the client authentication mechanism separately?

Motivation

This thesis proposes to ..

... implement FIDO2 authentication within the TLS handshake

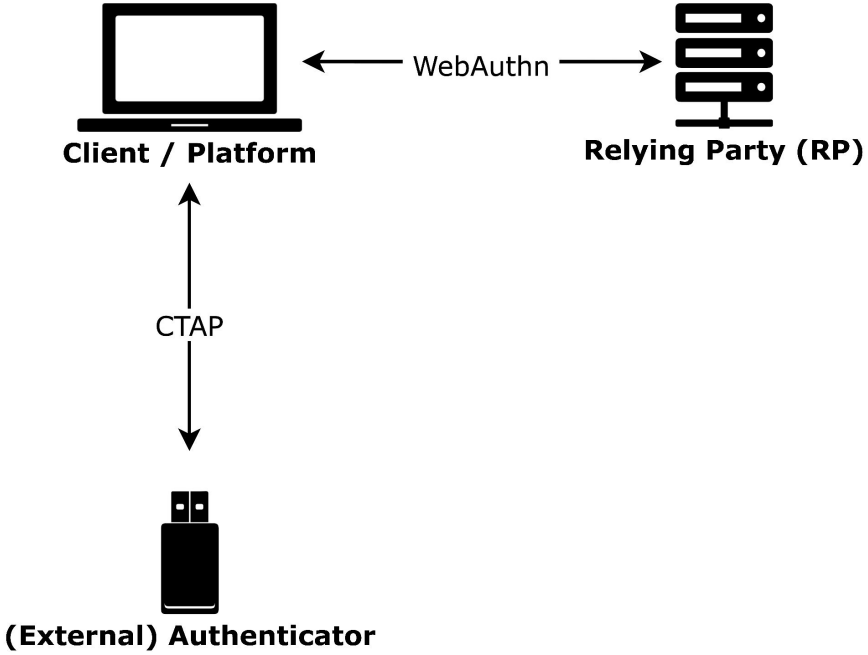
Hereby ...

.. generically adding FIDO2 client authentication **to all TLS-based protocols**

.. restoring the conceptual integrity of the OSI model

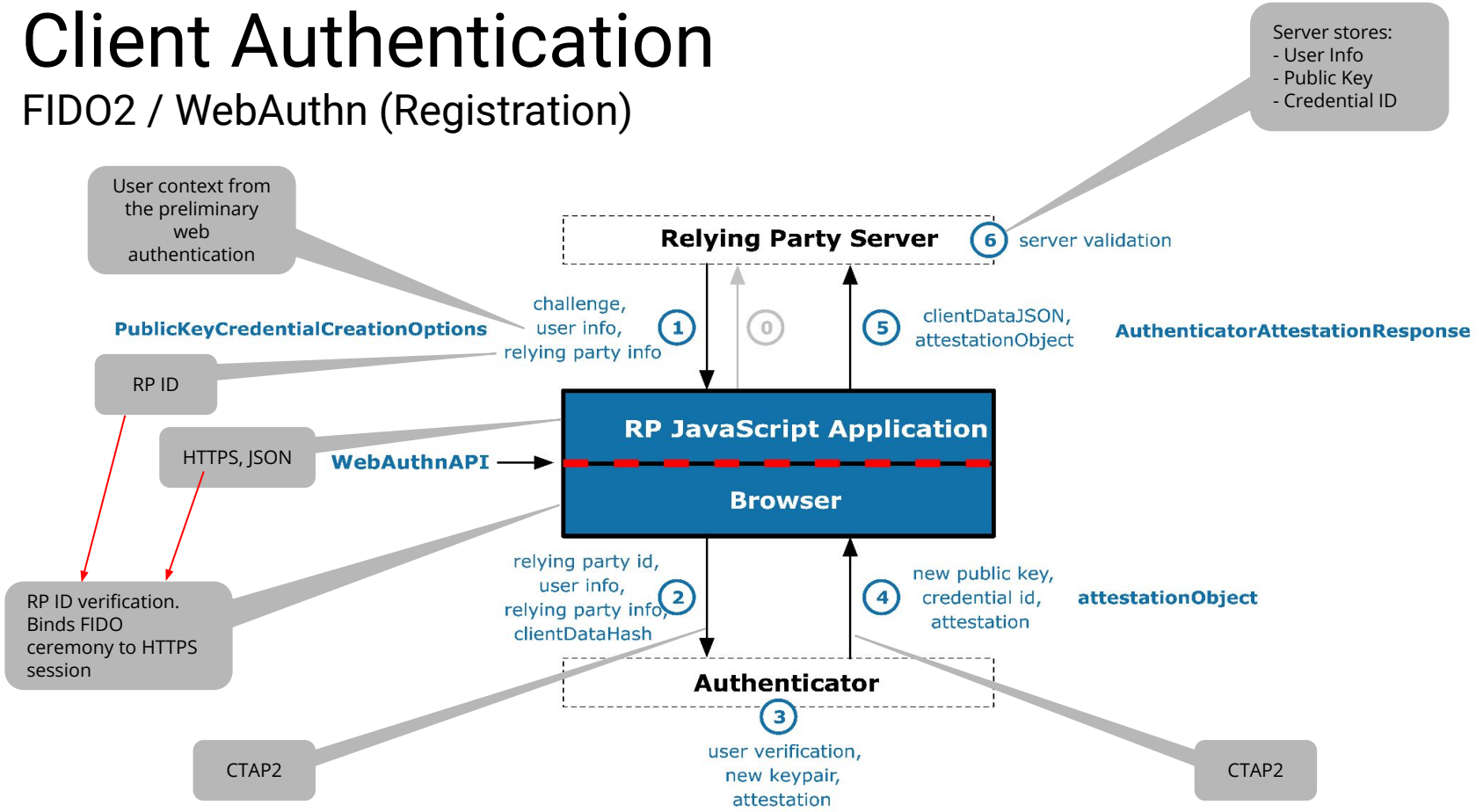
Client Authentication

FIDO2 / WebAuthn (Topology)



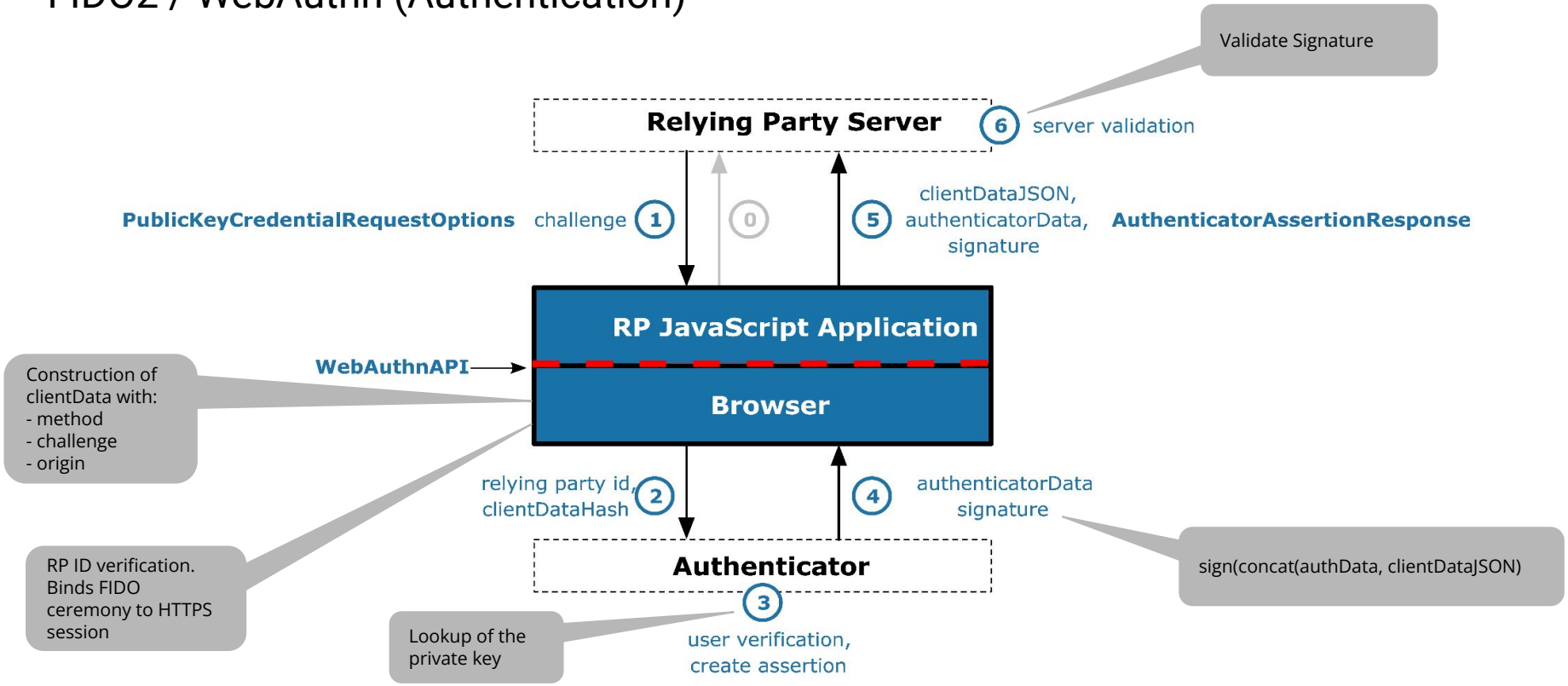
Client Authentication

FIDO2 / WebAuthn (Registration)



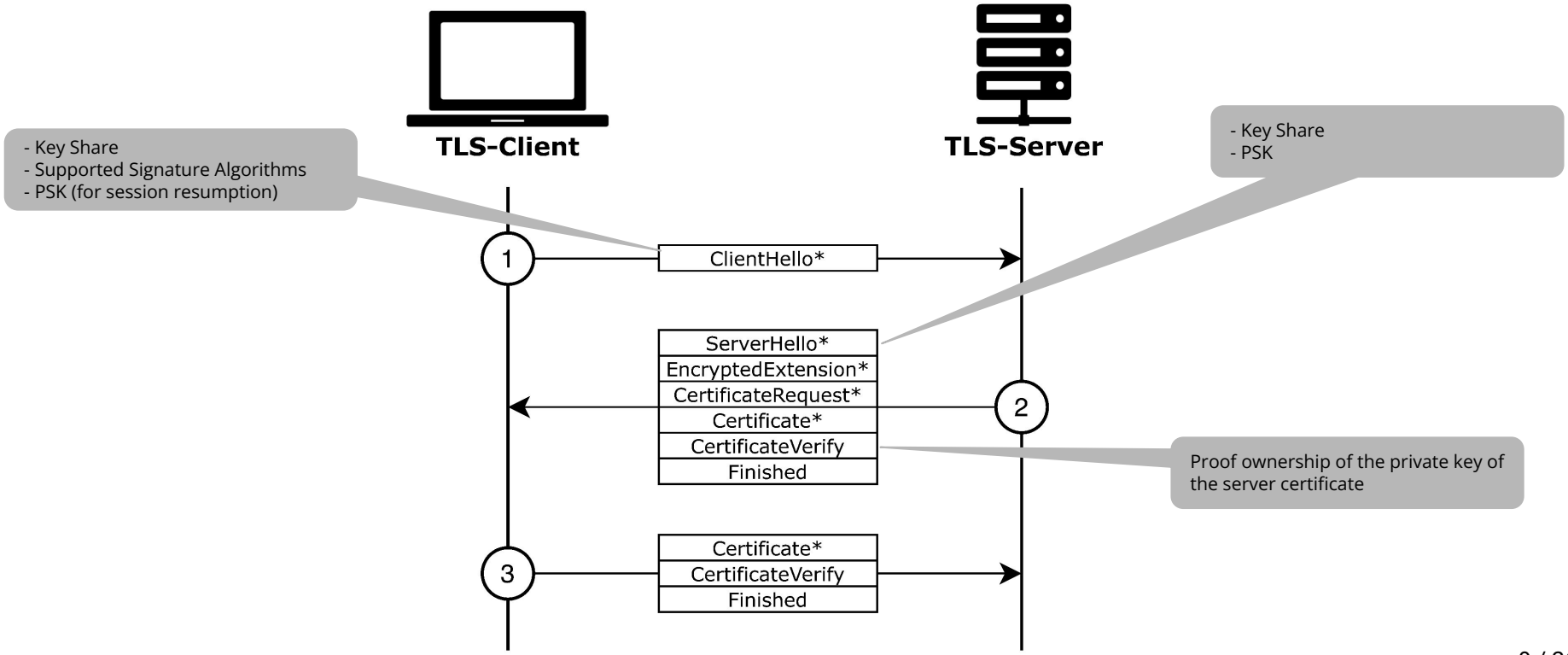
Client Authentication

FIDO2 / WebAuthn (Authentication)



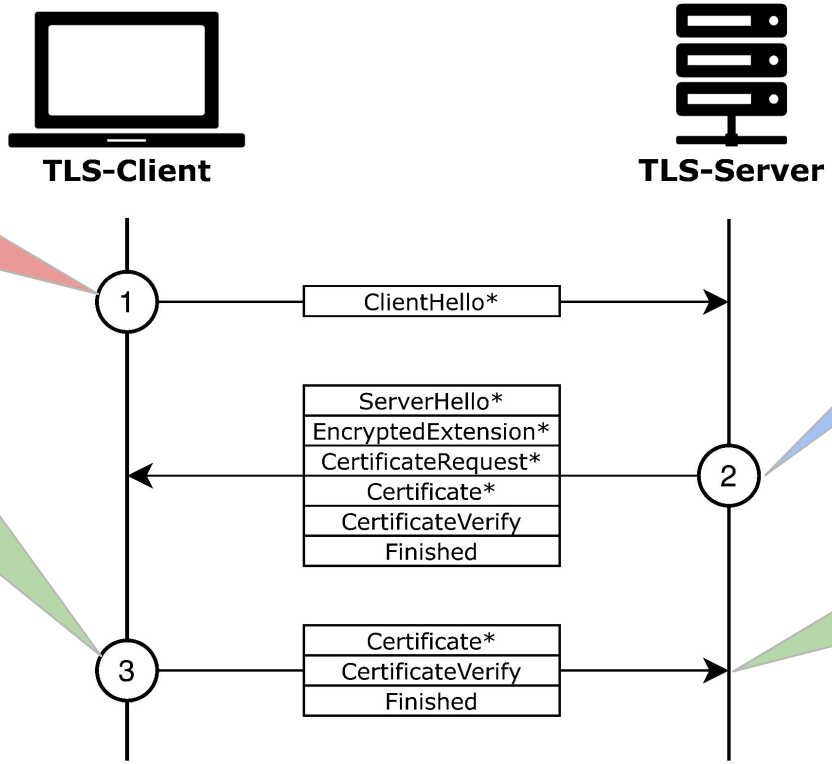
Client Authentication

TLS 1.3 (Handshake with Client Certificates)



Client Authentication

TLS 1.3 (Handshake with Client Certificates)



Announcing capabilities, public keys, extensions

→ **NOT CONFIDENTIAL**

Announcing servers public keys, derives session keys, encryption starts

→ **CONFIDENTIAL, server certificate NOT YET AUTHENTICATED**

Validates server cert, verifies servers signature, derives session keys, verifies Finished

→ **CONFIDENTIAL, SERVER AUTHENTICATED**

Checks CertificateVerify verifies Finished,

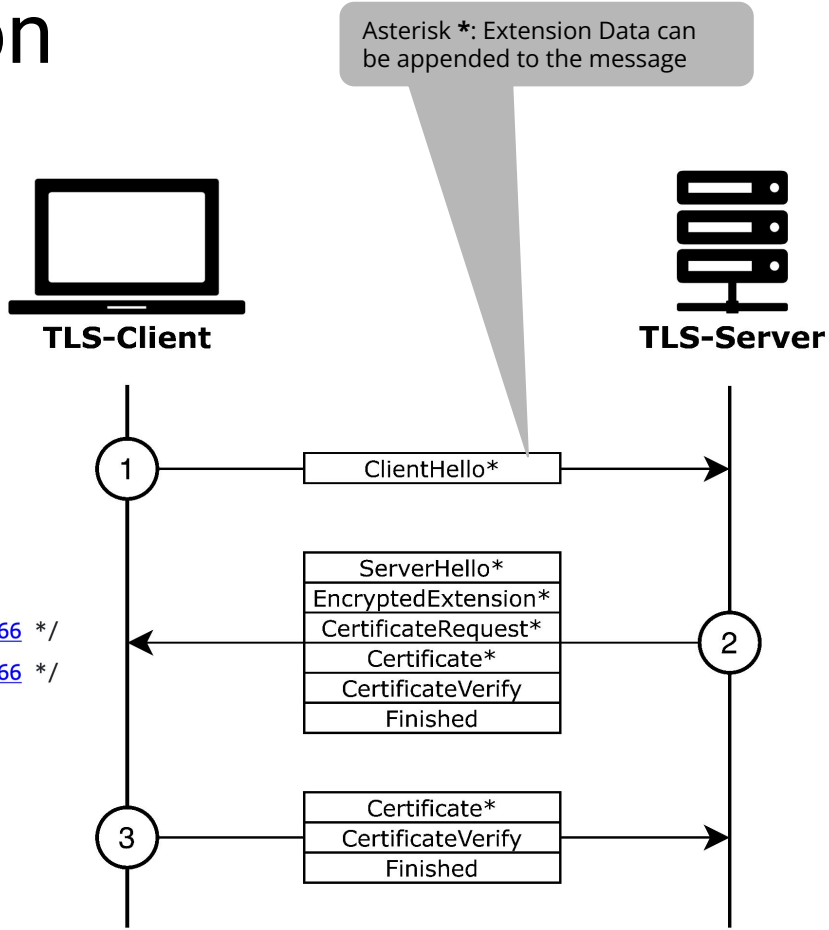
→ **CONFIDENTIAL, SERVER and CLIENT AUTHENTICATED, MitM PROTECTION**

Client Authentication

TLS 1.3 (Extensions)

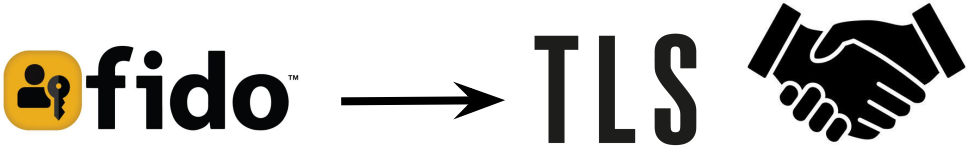
From RFC 8446: A number of TLS messages contain **tag-length-value encoded extensions structures**.

```
struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
} Extension;

enum {
    server_name(0),                     /* RFC 6066 */
    max_fragment_length(1),             /* RFC 6066 */
    ...

    (65535)
} ExtensionType;
```



FIDO2 as TLS 1.3 Extension

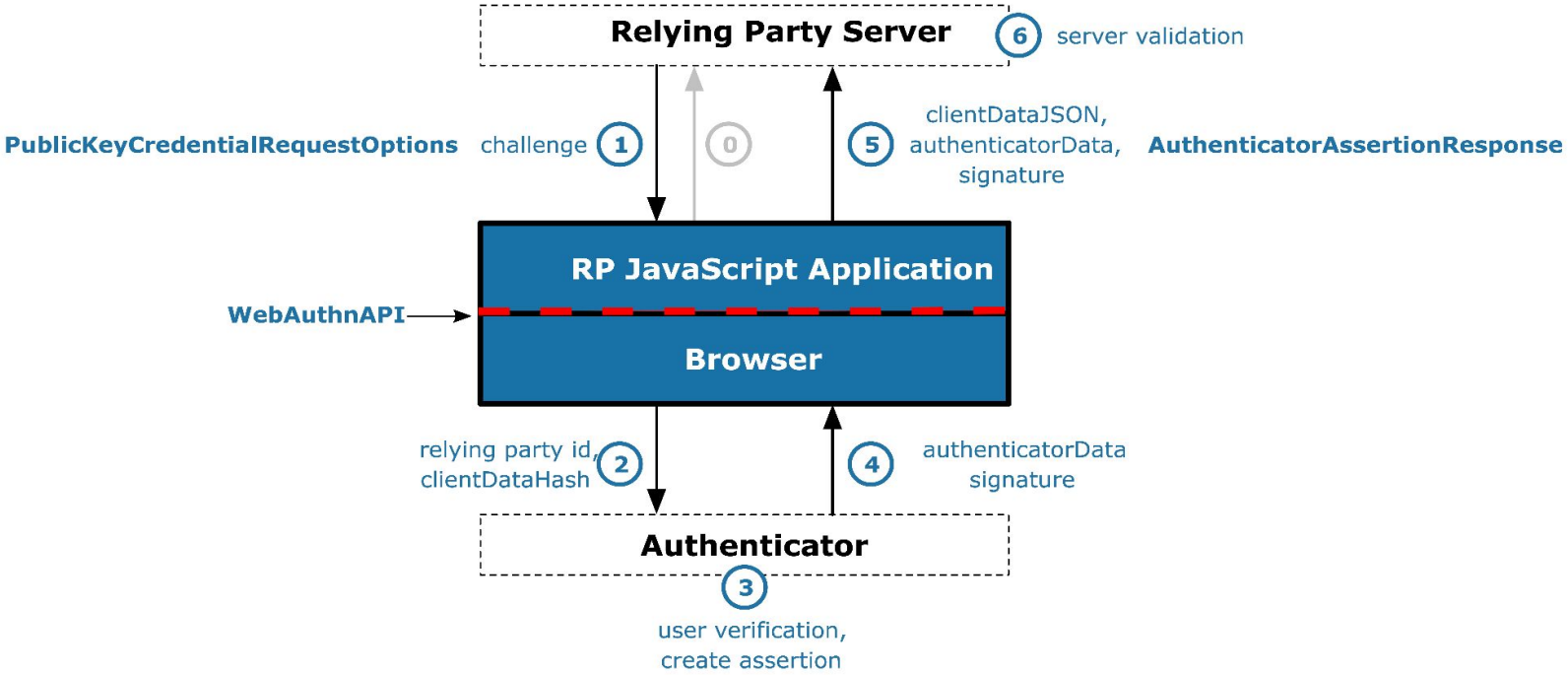
Design Principles



- FIDO-authentication (or registration) **still in TLS handshake**, before client sends his finished-message
  - Application data not interleaved with FIDO ceremony data
  - Application can query the TLS library for the outcome of the authentication
- TLS handshake is **not altered**. FIDO ceremony data is sent only as a **TLS extension payload**.

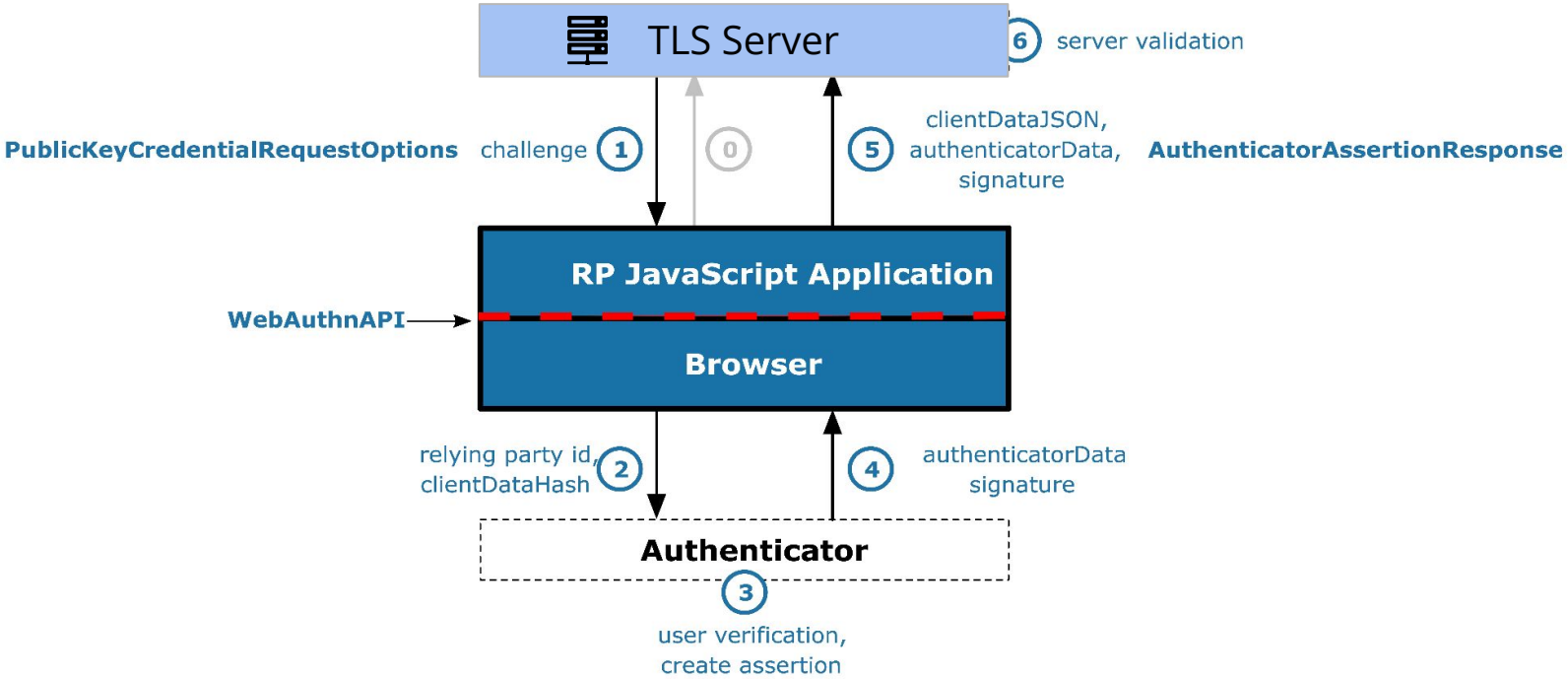
FIDO2 as TLS 1.3 Extension

Design Principles



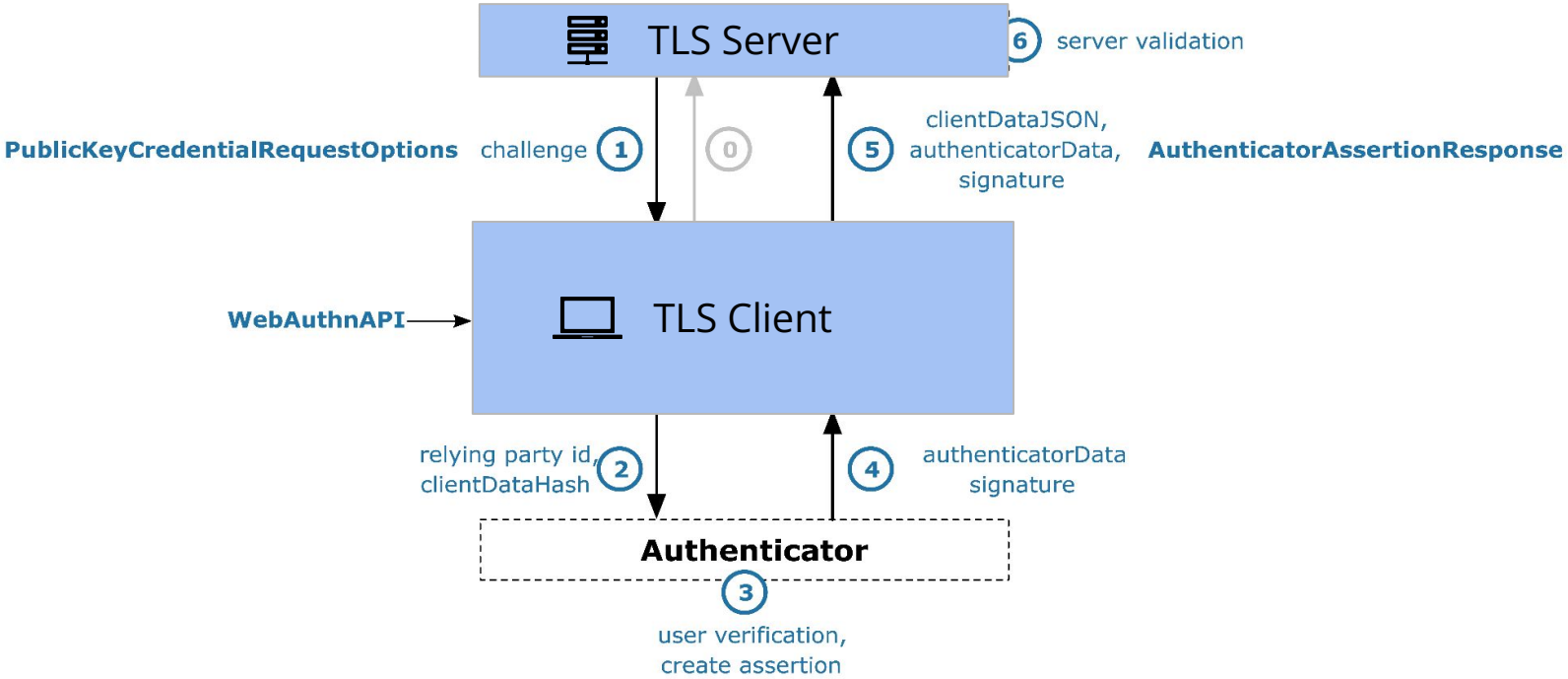
FIDO2 as TLS 1.3 Extension

Design Principles



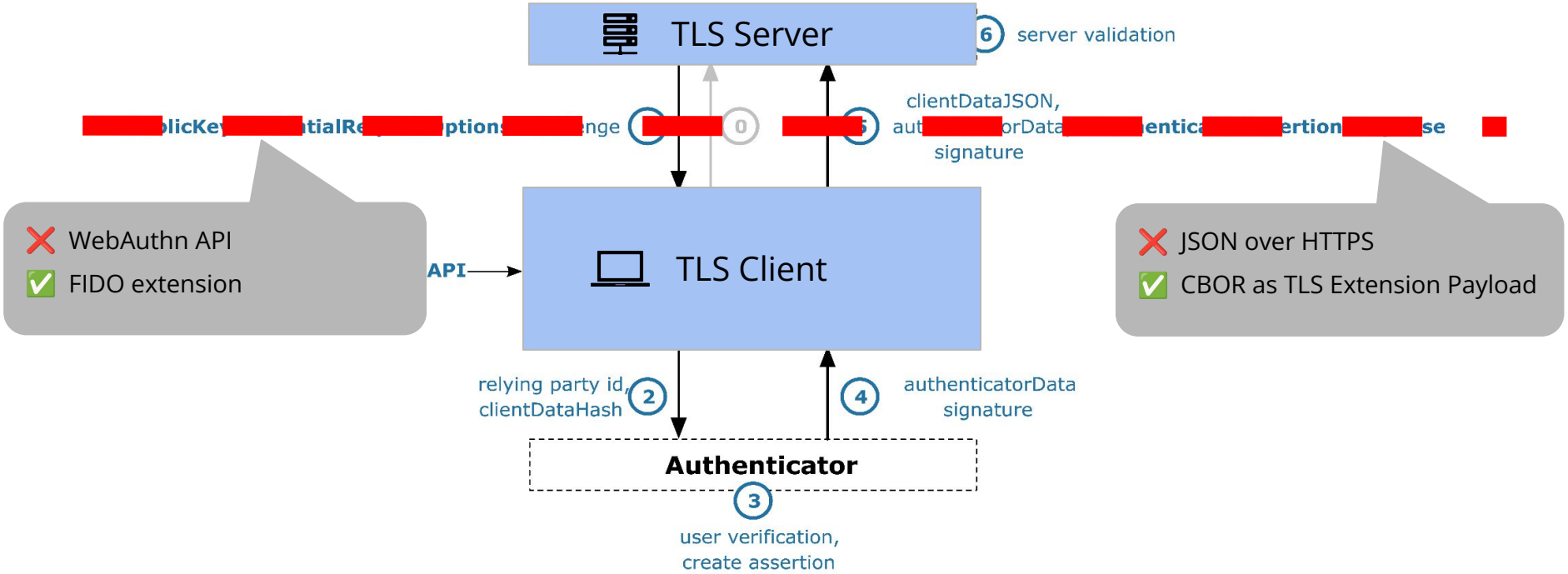
FIDO2 as TLS 1.3 Extension

Design Principles



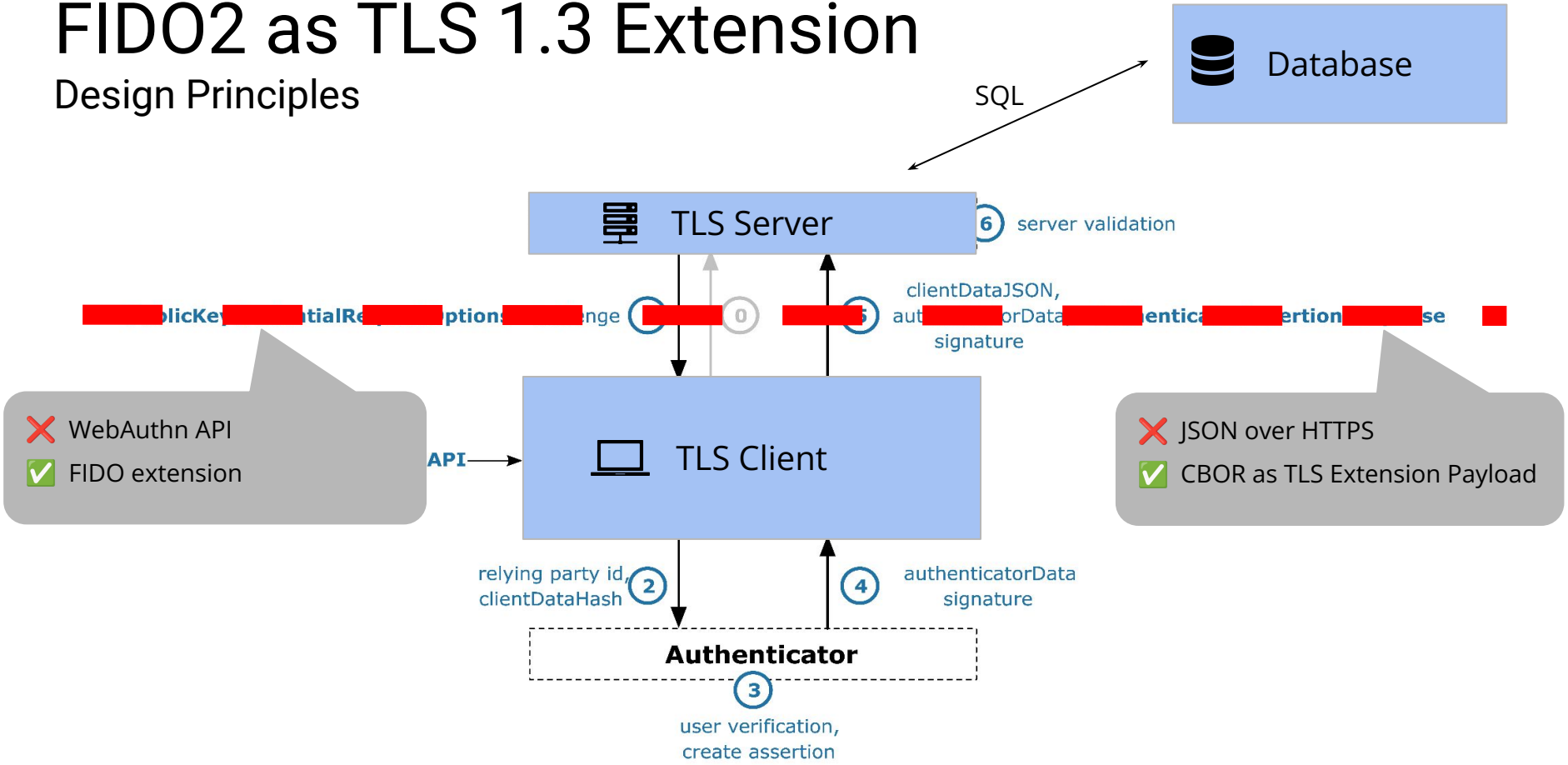
FIDO2 as TLS 1.3 Extension

Design Principles



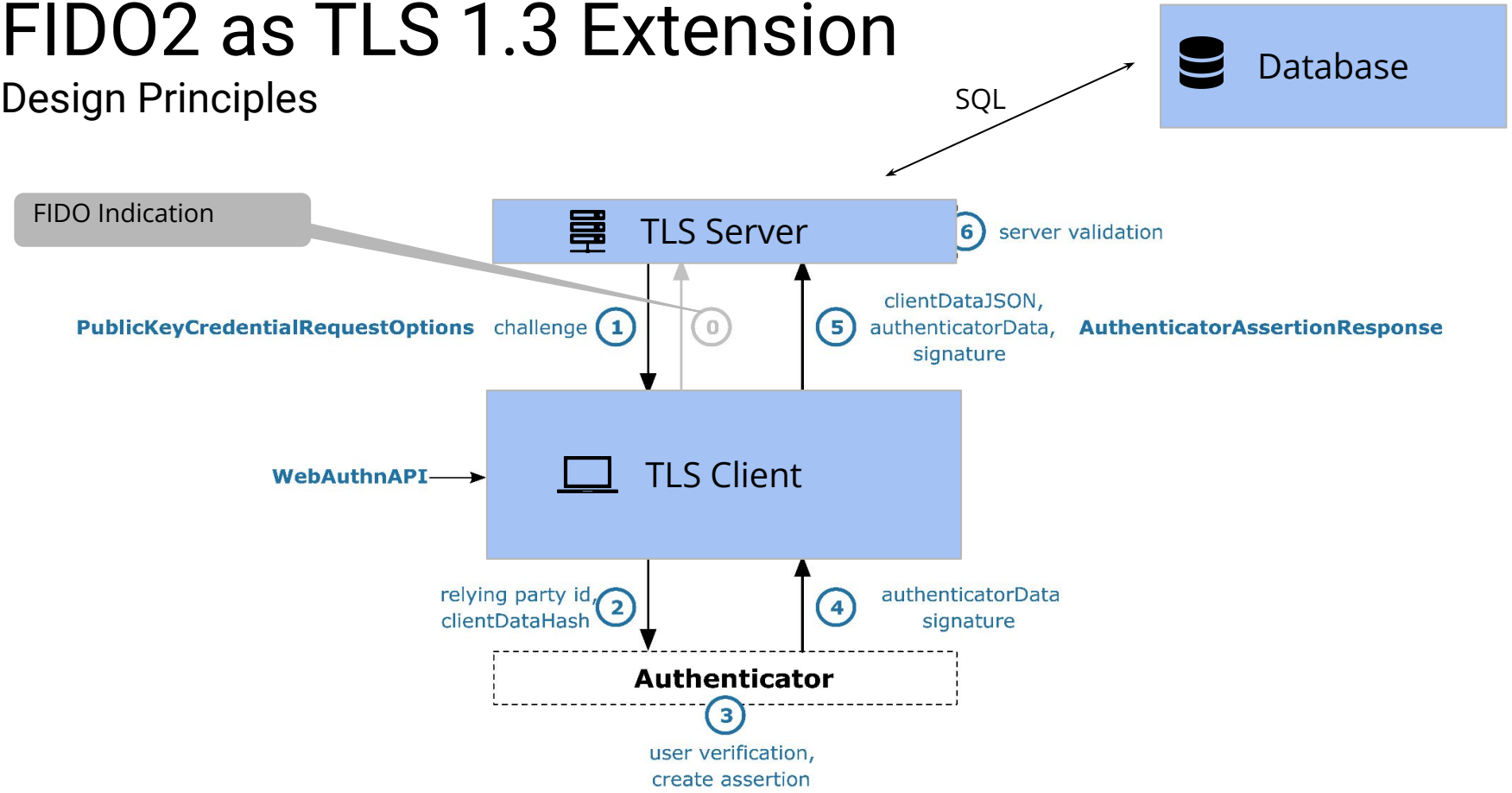
FIDO2 as TLS 1.3 Extension

Design Principles



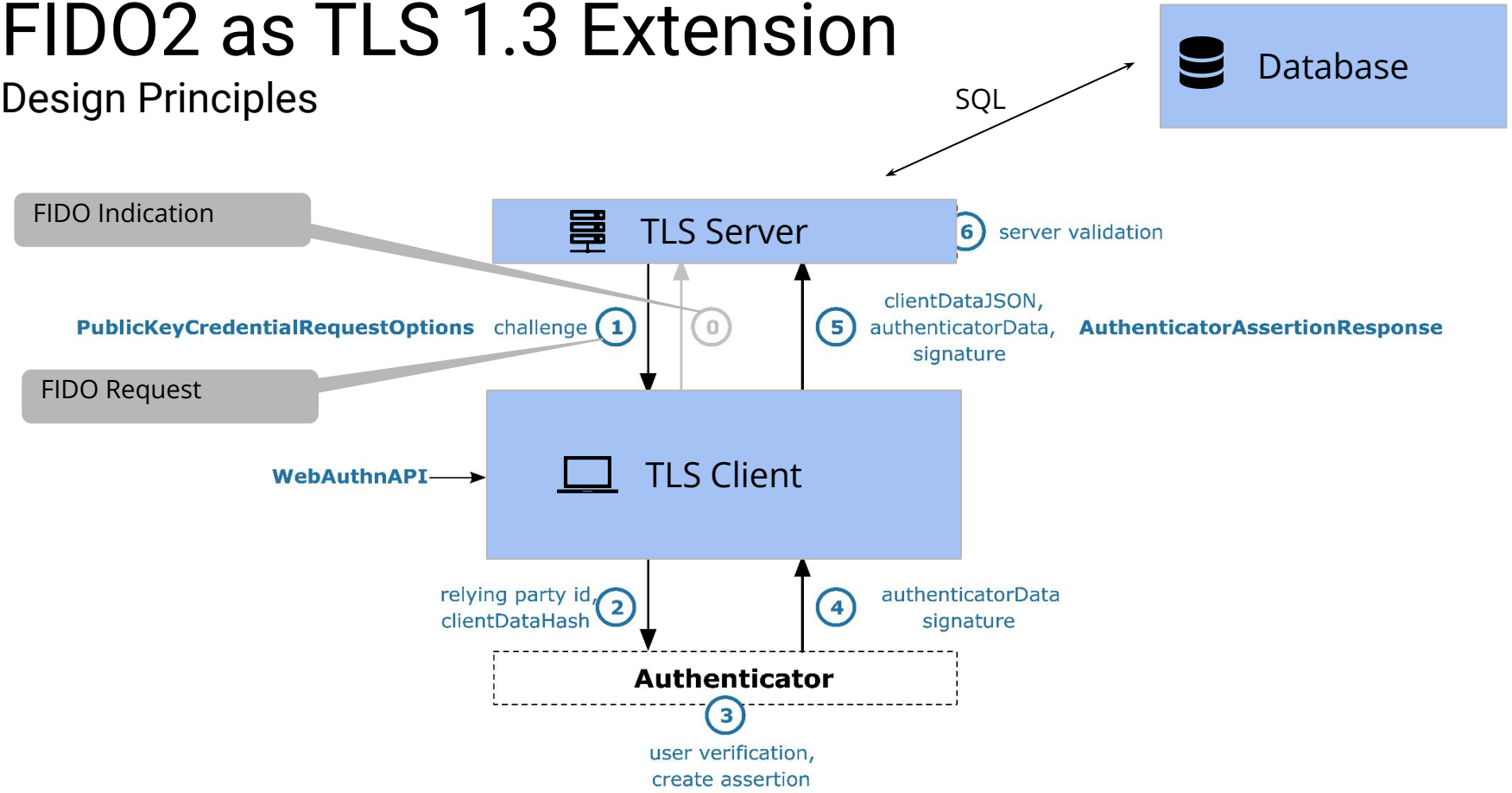
FIDO2 as TLS 1.3 Extension

Design Principles



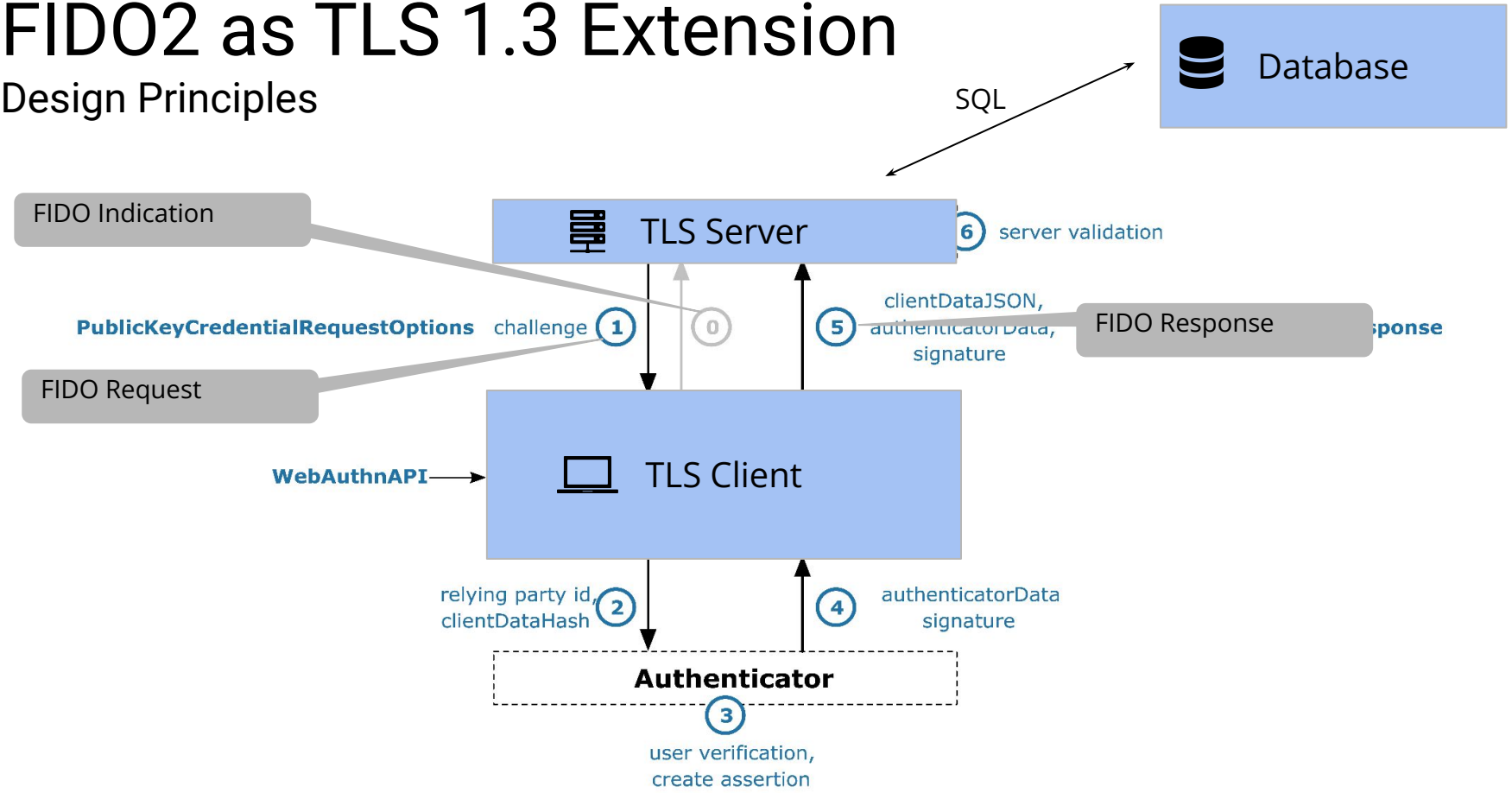
FIDO2 as TLS 1.3 Extension

Design Principles



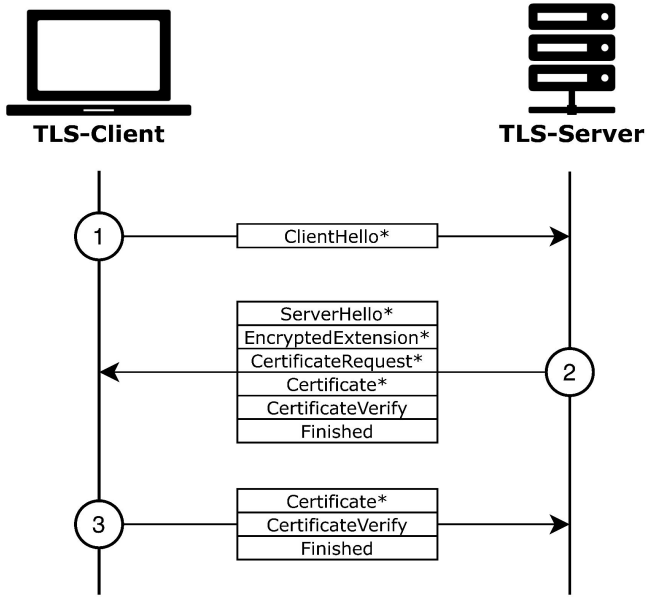
FIDO2 as TLS 1.3 Extension

Design Principles



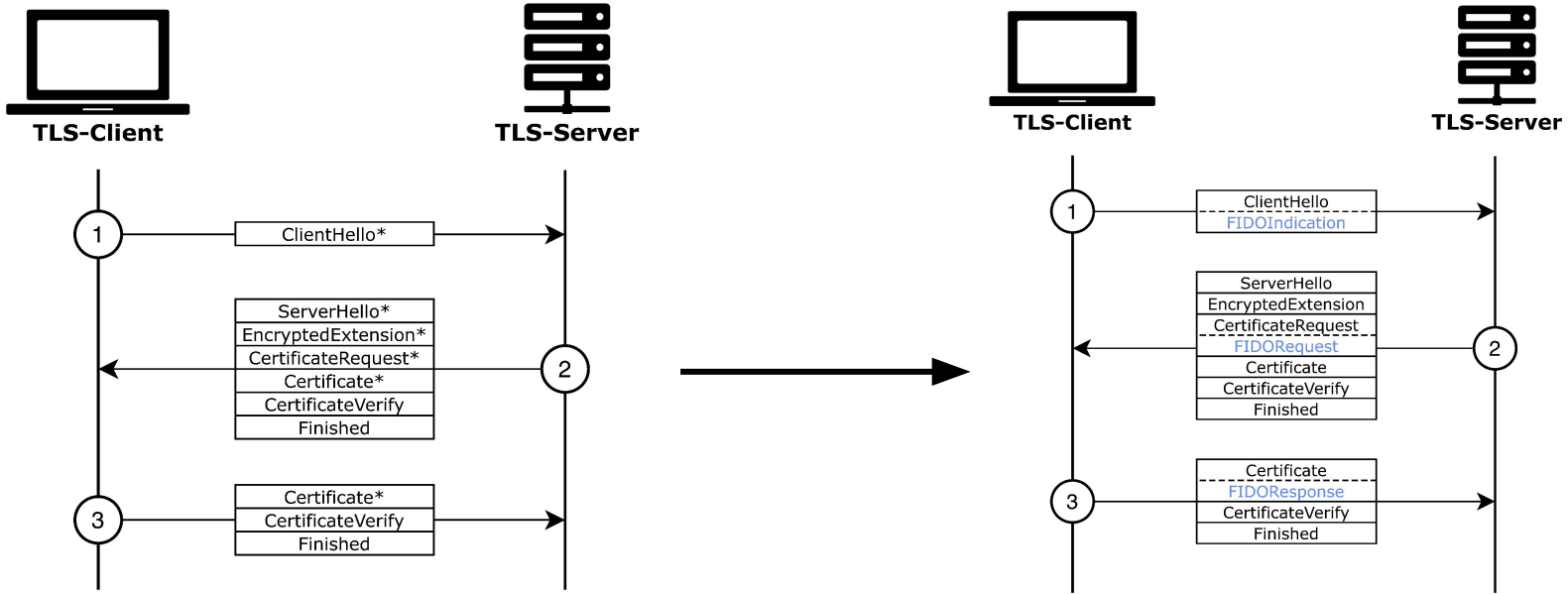
FIDO2 as TLS 1.3 Extension

Integration into TLS handshake



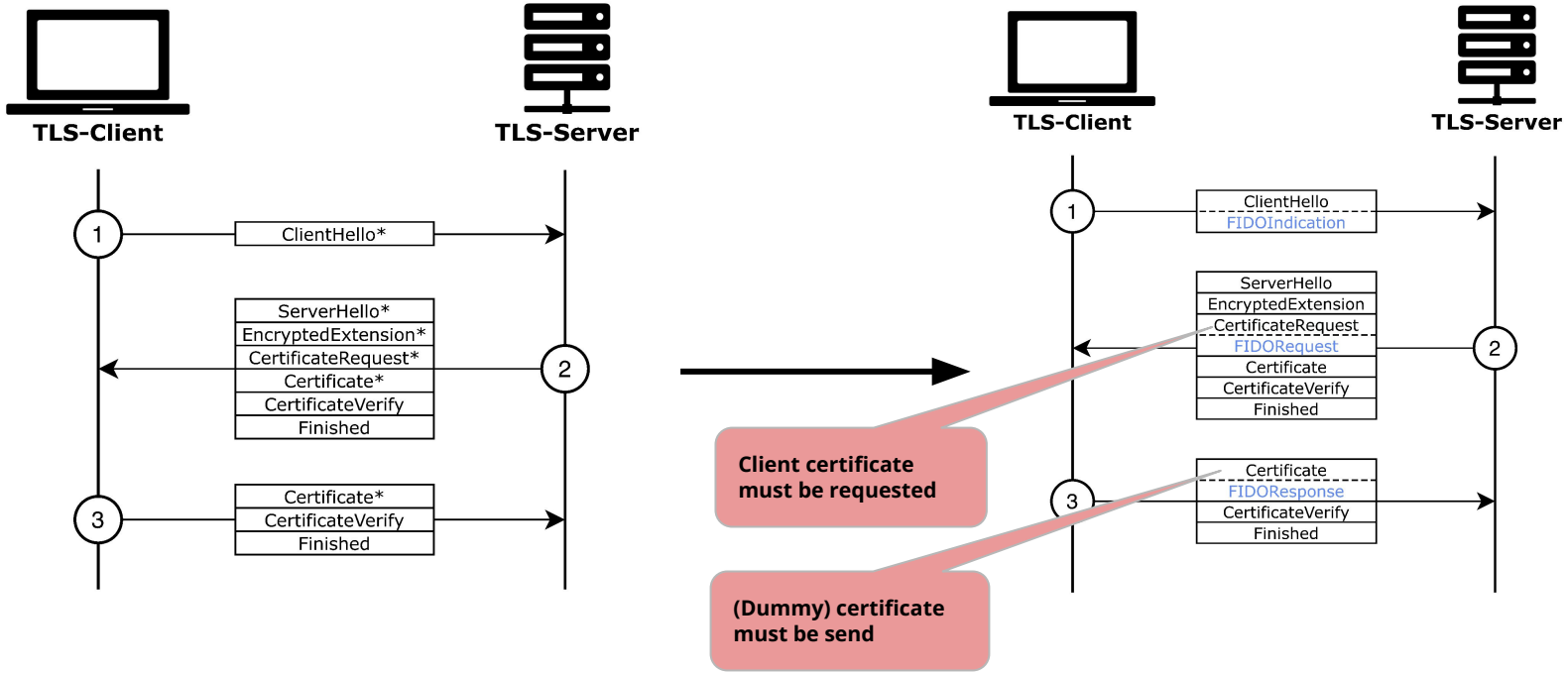
FIDO2 as TLS 1.3 Extension

Integration into TLS handshake



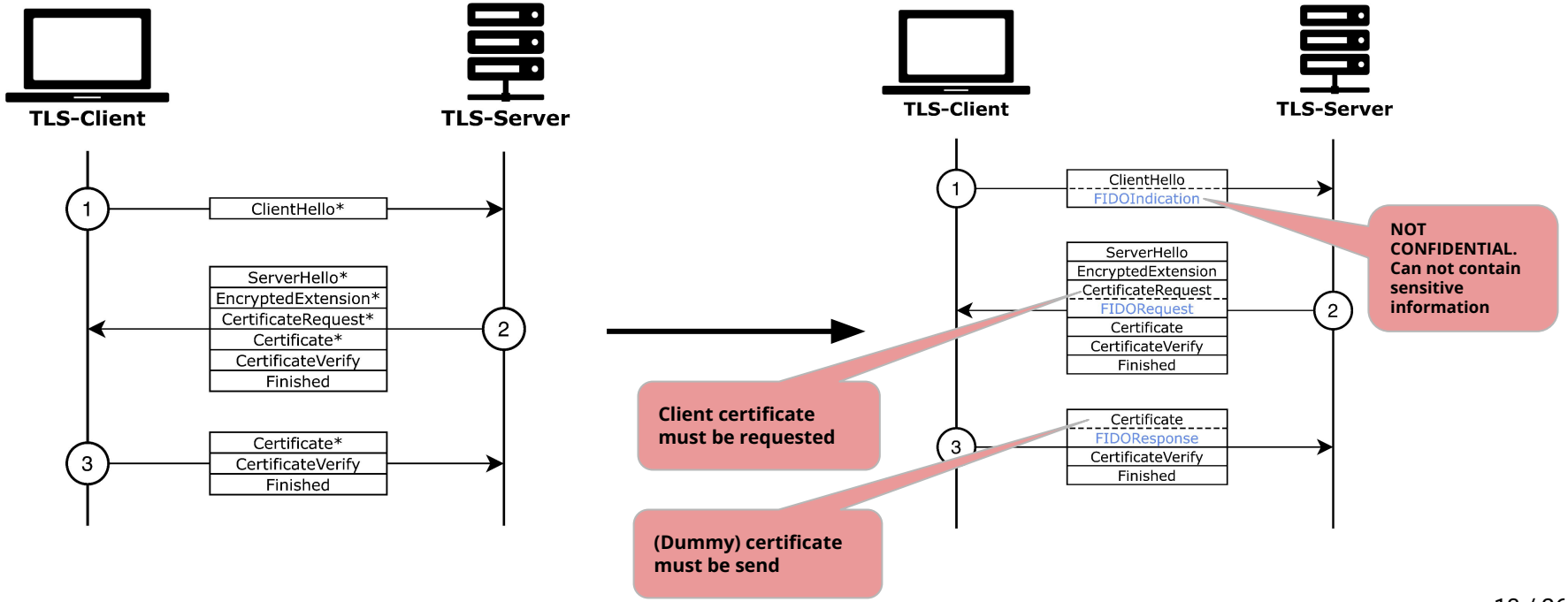
FIDO2 as TLS 1.3 Extension

Integration into TLS handshake



FIDO2 as TLS 1.3 Extension

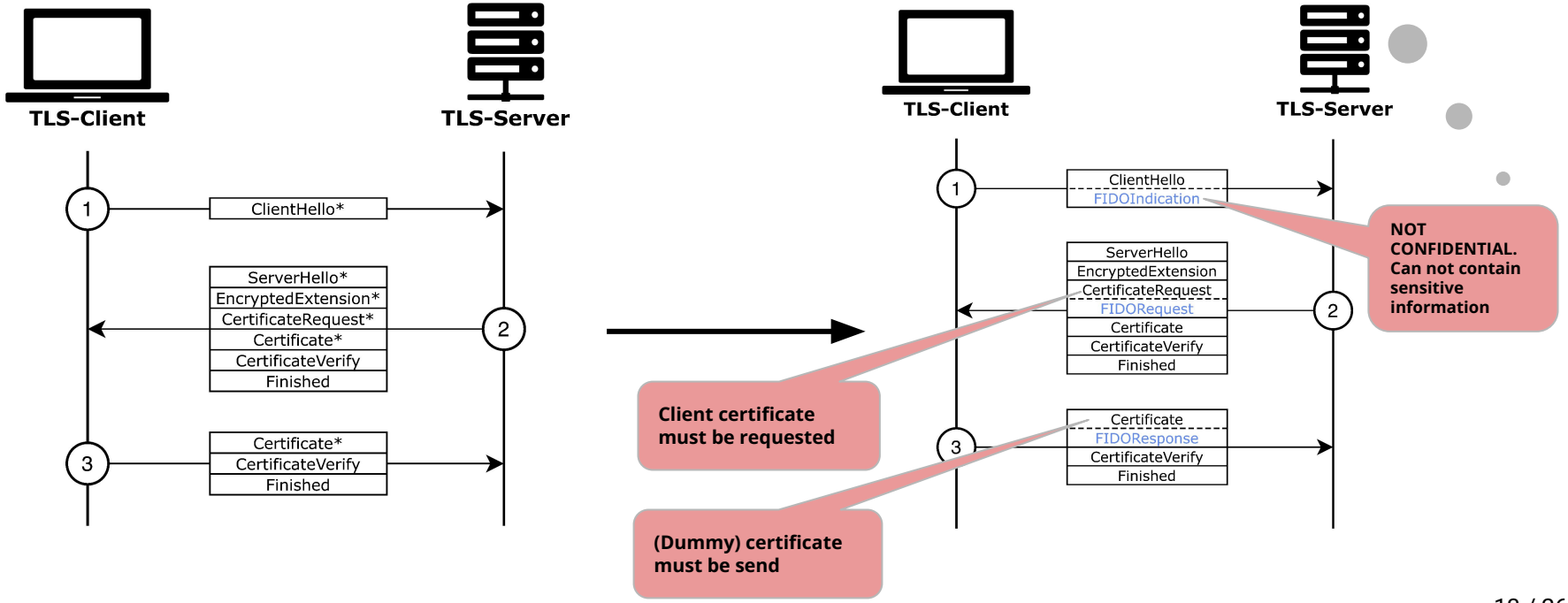
Integration into TLS handshake



FIDO2 as TLS 1.3 Extension

Integration into TLS handshake

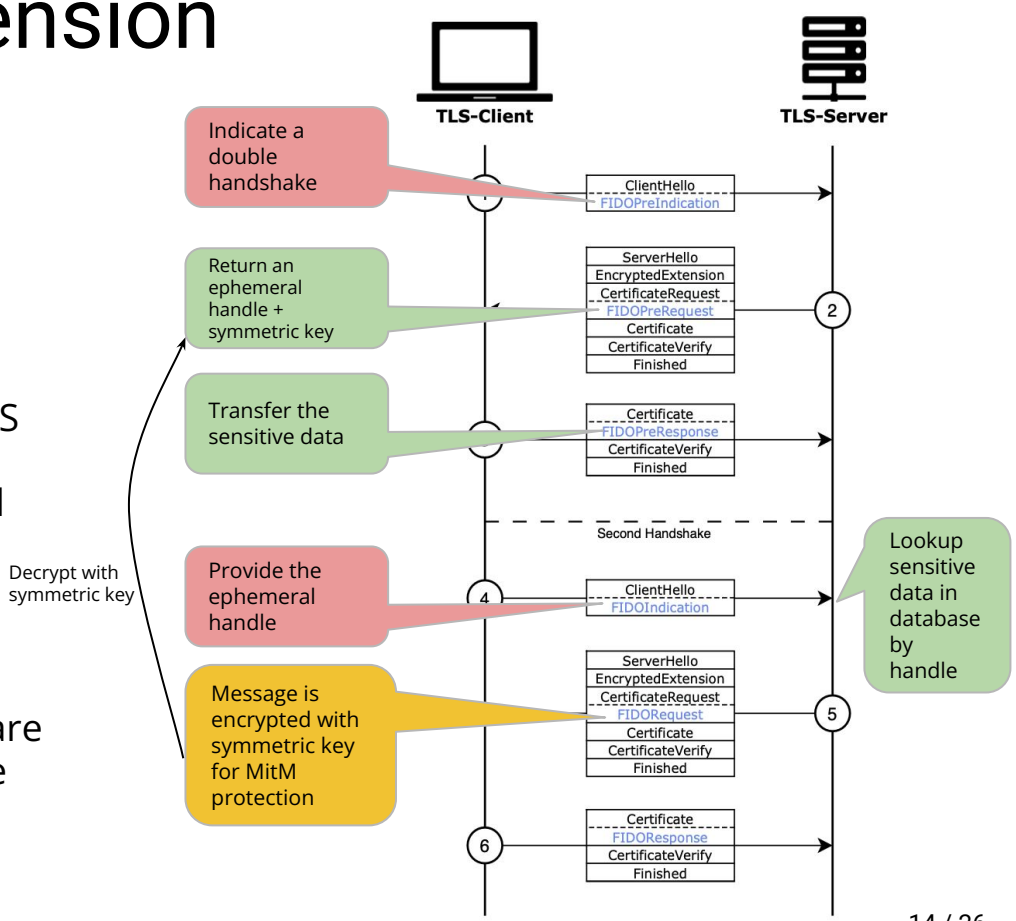
Work for authentication with discoverable credentials. What about non-discoverable creds and key registration?



FIDO2 as TLS 1.3 Extension

Integration into TLS handshake

- Idea: Sensitive data of the *FIDOIndication* (**user identity**) is transferred in a preceding TLS handshake.
  - After the first TLS handshake, the TLS connection is closed
  - The second handshake is performed over the same TCP connection
- Overhead rational:
  - Key registration ideally **only done once** or **out of band**.
  - Non-discoverable credentials (U2F) are deprecated in favour of discoverable credentials



FIDO2 as TLS 1.3 Extension

Message Structure & Encoding

Traditional FIDO:

- CTAP (Client ↔ Authenticator):
  CBOR encoded ✅
  - Space efficient representation
  - Easy to parse
- WebAuthn (RP ↔ Client):
  Serialized JSON objects ❌
  - Space-inefficient (one byte per curly brace)
  - Easy to read & debug

FIDO2 as TLS 1.3 Extension

Message Structure & Encoding

Traditional FIDO:

- CTAP (Client ↔ Authenticator): **CBOR** encoded ✅
  - Space efficient representation
  - Easy to parse
- WebAuthn (RP ↔ Client): **Serialized JSON objects** ❌
  - Space-inefficient (one byte per curly brace)
  - Easy to read & debug

FIDO TLS extension:

- CTAP (Client ↔ Authenticator): **CBOR** encoded ✅
  - Space efficient representation
  - Easy to parse
- Extension Data (TLS-Server ↔ TLS-Client): **CBOR** ✅
  - Extensive documentation in the thesis
  - A standard would be desirable for interoperability

FIDO2 as TLS 1.3 Extension

M Encoding

Tr

- cator):

ation

- WebAuthn (RP ↔ Client):

  Serialized JSON objects ❌

  - Space-inefficient (one byte per curly brace)
  - Easy to read & debug

A.2. Pre-Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 2
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Request.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference that links this handshake to the following one.
- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetric encryption, utilizing the AES algorithm in conjunction with GCM.

int	binary	binary
MESSAGE TYPE	EPHEMERAL USER ID	GCM KEY

FIDO TLS extension:

- CTAP (Client ↔ Authenticator): **CBOR** encoded ✅
  - Space efficient representation
  - Easy to parse
- Extension Data (TLS-Server ↔ TLS-Client): **CBOR** ✅
  - Extensive documentation in the thesis
  - A standard would be desirable for interoperability

FIDO2 as TLS 1.3 Extension

M… Encoding

Tr…

- …cator):
  - …ation
- WebAuthn (RP ↔ Client):

  Serialized JSON objects ❌
  - Space-inefficient (one byte per curly brace)
  - Easy to read & debug

A.2. Pre-Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 2
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Request.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference that links this handshake to the following one.
- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetric encryption, utilizing the AES algorithm in conjunction with GCM.

int	binary	binary
MESSAGE TYPE	EPHEMERAL USER ID	GCM KEY

FIDO TLS ext…

- CTAP (Cl…

  CBOR er…
  - Space efficient representation
  - Easy to parse
- Extension Data (TLS-Server ↔ TLS-Client): **CBOR ✅**
  - Extensive documentation in the thesis
  - A standard would be desirable for interoperability

A.3. Pre-Registration Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 3
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Response.
- **USER NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The username associated with the end user's account.
- **USER DISPLAY NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The full name of the user, intended for display purposes within user.
- **TICKET**:
  Type: Byte String
  Length: 128 – 512 Bytes
  Description: A unique ticket that enabled the user to register new FIDO keys with the RP.

int	string	string	binary
MESSAGE TYPE	USER NAME	USER DISPLAY NAME	TICKET

FIDO2 as TLS 1.3 Extension

M… Encoding

Tr…

- …cator):
  - …ation
- WebAuthn (RP ↔ C…
  Serialized JSON di…
  - Space-inefficient (o… brace)
  - Easy to read & deb…

FIDO TLS ext…

- CTAP (Cl…
  CBOR er…
  - Space efficient representation
    … parse
  - … Data (TLS-Server ↔
    …): **CBOR** ✅
    …ve documentation in the thesis
    … be integrated into a standard
    …roperability

A.2. Pre-Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 2
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Request.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference that links this handshake to the following one.
- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetric encryption, utilizing the AES algorithm in conjunction with GCM.

int	binary	binary
MESSAGE TYPE	EPHEMERAL USER ID	GCM KEY

A.3. Pre-Registration Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 3
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Response.
- **USER NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The username associated with the end user's account.
- **USER DISPLAY NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The full name of the user, intended for display purposes within user.
- **TICKET**:
  Type: Byte String
  Length: 128 – 512 Bytes
  Description: A unique ticket that enabled the user to register new FIDO keys with the RP.

int	string	string	binary
MESSAGE TYPE	USER NAME	USER DISPLAY NAME	TICKET

A.4. Registration Indication

- **MESSAGE TYPE**:
  Type: Integer
  Value: 4
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Registration Indication.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference to a previous handshake.

int	binary
MESSAGE TYPE	EPHEMERAL USER ID

FIDO2 as TLS 1.3 Extension

M Encoding

Tr Fl

- cator):

ation resentation

- WebAuthn (RP ↔ C S-Server ↔

Serialized JSON di ✅

- Space-inefficient (o ve documentation in the thesis brace)
- Easy to debug be integrated into a standard roperability

A.2. Pre-Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 2
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Request.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference that links this handshake to the following one.
- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetric encryption, utilizing the AES algorithm in conjunction with GCM.

int	binary	binary
MESSAGE TYPE	EPHEMERAL USER ID	GCM KEY

A.3. Pre-Registration Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 3

ssage, indicating a Pre-Registration Response.

d with the end user's account.

er, intended for display purposes within user.

abled the user to register new FIDO keys with

string	binary
USER DISPLAY NAME	TICKET

A.4. Registration Indication

- **MESSAGE TYPE**:
  Type: Integer
  Value: 4
  Length: 1 Byte
  Description: Specifies the type of message, indicati
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference to a previous handshake.

int	binary
MESSAGE TYPE	EPHEMERAL USER ID

A.5. Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 5
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Registration Request.
- **CHALLENGE**:
  Type: Byte String
  Length: 16 – 64 Bytes
  Description: Random challenge from the RP.
- **RP ID**:
  Type: UTF-8 String
  Length: Variable, up to 256 Bytes
  Description: Relying Party identifier.
- **RP NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: A human-readable name for the Relying Party, used mainly for display.
- **USER NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The username associated with the end user's account. This value is encrypted with the GCM Key.
- **USER DISPLAY NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The full name of the user, intended for display purposes within user. This value is encrypted with the GCM Key.
- **USER ID**:
  Type: Byte String
  Length: Variable, up to 64 Bytes
  Description: A unique identifier for the user in the context of the Relying Party. This value is encrypted with the GCM Key.
- **PUBKEY CRED PARAMS**:
  Type: Array of Integers
  Length: 1 – 6 Bytes
  Description: List of possible public key algorithms, in descending preference. Each

FIDO2 as TLS 1.3 Extension

M Encoding

Tr

- WebAuthn

Serialized

- Space-in
  brace)
- Easy to c

Fl

resentation

S-Server ↔

✅

ve documentation in the thesis

be integrated into a standard

roperability

A.2. Pre-Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 2
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Request.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference t
- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetri
  conjunction with GCM.

int	binary
MESSAGE TYPE	EPHEME USER

Integer represents a enum value, mapped to following public key algorithms:
0: COSE_ES256
1: COSE_ES384
2: COSE_EDDSA
3: COSE_ECDH_ES256
4: COSE_RS256
5: COSE_RS1

- **OPTIONALS**:
  Type: CBOR map
  Description: Contains optional values that can be probed by their keys.
  - **TIMEOUT**:
    Type: Integer
    Key: 1
    Length: 4 Bytes
    Description: Maximum time, in milliseconds, that the client should wait for the user to complete the action.
  - **AUTHENTICATOR SELECTION**:
    Type: CBOR array
    Key: 2
    Description: Criteria the RP wants to impose regarding the authenticators to be used. All values of the array are encrypted with the GCM key.
    - **ATTACHMENT**:
      Type: Integer
      Length: 1 Byte
      Description: Criteria the RP wants to impose regarding authenticator attachment. Enum value that maps to:
      0: PLATFORM
      1: CROSS-PLATFORM
    - **RESIDENT KEY**:
      Type: Integer
      Length: 1 Byte
      Description: Criteria the RP wants to impose regarding discoverable credentials. Enum value that maps to:
      0: REQUIRED
      1: PREFERRED
      2: DISCOURAGED
    - **USER VERIFICATION**:
      Type: Integer
      Length: 1 Byte
      Description: Criteria the RP wants to impose regarding user verification. Enum value that maps to:

ion

type of message, indicati

ID:

6 Bytes

reference to a previous handshake.

int	binary
MESSAGE TYPE	EPHEMERAL USER ID

A.3. Pre-Registration Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 3

ssage, indicating a Pre-Registration Response.

d with the end user's account.

er, intended for display purposes within user.

bled the user to register new FIDO keys with

string	binary
USER DISPLAY NAME	TICKET

A.5. Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 5
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Registration Request.
- **CHALLENGE**:
  Type: Byte String
  Length: 16 – 64 Bytes
  Description: Random challenge from the RP.
- **RP ID**:
  Type: UTF-8 String
  Length: Variable, up to 256 Bytes
  Description: Relying Party identifier.
- **RP NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: A human-readable name for the Relying Party, used mainly for display.
- **USER NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The username associated with the end user's account. This value is encrypted with the GCM Key.
- **USER DISPLAY NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The full name of the user, intended for display purposes within user. This value is encrypted with the GCM Key.
- **USER ID**:
  Type: Byte String
  Length: Variable, up to 64 Bytes
  Description: A unique identifier for the user in the context of the Relying Party. This value is encrypted with the GCM Key.
- **PUBKEY CRED PARAMS**:
  Type: Array of Integers
  Length: 1 – 6 Bytes
  Description: List of possible public key algorithms, in descending preference. Each

FIDO2 as TLS 1 … on

M…

Tr…

- …

- WebAuthn … S-Server ↔

Serialized … ✅

- Space-in… ve documentation in the thesis brace) … be integrated into a standard
- Easy to … roperability

A.2. Pre-Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 2
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Request
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference t…
- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetri… conjunction with GCM.

int	binary
MESSAGE TYPE	EPHEME… USER …

Integer represents a enum value, mapped to follo…
0: COSE_ES256
1: COSE_ES384
2: COSE_EDDSA
3: COSE_ECDH_ES256
4: COSE_RS256
5: COSE_RS1

- **OPTIONALS**:
  Type: CBOR map
  Description: Contains optional values that can b…
  - **TIMEOUT**:
    Type: Integer
    Key: 1
    Length: 4 Bytes
    Description: Maximum time, in millisecond… the user to complete the action.
  - **AUTHENTICATOR SELECTION**:
    Type: CBOR array
    Key: 2
    Description: Criteria the RP wants to impos… be used. All values of the array are encrypt…
    - **ATTACHMENT**:
      Type: Integer
      Length: 1 Byte
      Description: Criteria the RP wants to impose regarding authenticator attachment. Enum value that maps to:
      0: PLATFORM
      1: CROSS-PLATFORM
    - **RESIDENT KEY**:
      Type: Integer
      Length: 1 Byte
      Description: Criteria the RP wants to impose regarding discoverable credentials. Enum value that maps to:
      0: REQUIRED
      1: PREFERRED
      2: DISCOURAGED
    - **USER VERIFICATION**:
      Type: Integer
      Length: 1 Byte
      Description: Criteria the RP wants to impose regarding user verification. Enum value that maps to:
      0: REQUIRED
      1: PREFERRED
      2: DISCOURAGED
  - **EXCLUDED CREDENTIALS**:
    Type: Nested N × 3 CBOR array
    Key: 3
    Description: This array contains credentials already registered with the user. If the authenticator detects any of these credentials as existing on the device, it must return an error to prevent the creation of duplicates. All values of the array are encrypted with the GCM key.
    - **CREDENTIAL TYPE**: *Type*: Integer
      Length: 1 Byte
      Description: Specifies the type of credential. Currently, only one type is supported by Webauthn, but the design is structured to allow for future expansions. Enum value that maps to:
      0: PUBLIC_KEY
    - **CREDENTIAL ID**:
      Type: Byte String
      Length: Variable, up to 256 Bytes
      Description: Unique identifier for a FIDO credential
    - **TRANSPORTS**:
      Type: Integer
      Length: 1 Byte
      Description: Mode of transportation for this credential. Enum value that maps to:
      0: USB
      1: NFC
      2: BLE
      3: INTERNAL
  - **ATTESTATION**:
    Type: Integer
    Key: 4
    Length: 1 Byte
    Description: Specifies the desired attestation conveyance preference of the RP. This values is encrypted with the GCM key. Enum value that maps to:
    0: NONE
    1: INDIRECT
    2: DIRECT
    3: ENTERPRISE
  - **EXTENSIONS**:
    Type: Nested M × 2 CBOR array
    Key: 5

…istration Request

…SAGE TYPE:
…Integer
…: 5
…h: 1 Byte
…ription: Specifies the type of message, indicating a Registration Request.

…LLENGE:
…Byte String
…h: 16 – 64 Bytes
…ription: Random challenge from the RP.

…D:
…UTF-8 String
…h: Variable, up to 256 Bytes
…ription: Relying Party identifier.

…NAME:
…UTF-8 String
…h: Variable, up to 255 Bytes
…ription: A human-readable name for the Relying Party, used mainly for display.

…R NAME:
…UTF-8 String
…h: Variable, up to 255 Bytes
…ription: The username associated with the end user's account. This value is …pted with the GCM Key.

…R DISPLAY NAME:
…UTF-8 String
…h: Variable, up to 255 Bytes
…ription: The full name of the user, intended for display purposes within user. …value is encrypted with the GCM Key.

…R ID:
…Byte String
…Length: Variable, up to 64 Bytes
Description: A unique identifier for the user in the context of the Relying Party. This value is encrypted with the GCM Key.

- **PUBKEY CRED PARAMS**:
  Type: Array of Integers
  Length: 1 – 6 Bytes
  Description: List of possible public key algorithms, in descending preference. Each

…type of message, indicati…
…ID:
…6 Bytes
…reference to a previous handshake.

int	binary
MESSAGE TYPE	EPHEMERAL USER ID

A.3. Pre-Registration Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 3

…ssage, indicating a Pre-Registration Response.

…ed with the end user's account.

…ser, intended for display purposes within user.

…abled the user to register new FIDO keys with

string	binary
USER DISPLAY NAME	TICKET

FIDO2 as TLS 1 … on

M…

Tr…

A.2. Pre-Registration Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 2
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Registration Request
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference t…
- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetri…
  conjunction with GCM.

Integer represents a enum value, mapped to follo…
0: COSE_ES256
1: COSE_ES384
2: COSE_EDDSA
3: COSE_ECDH_ES256
4: COSE_RS256
5: COSE_RS1

- **OPTIONALS**:
  Type: CBOR map
  Description: Contains optional values that can b…
  – **TIMEOUT**:

0: REQUIRED
1: PREFERRED
2: DISCOURAGED

– **EXCLUDED CREDENTIALS**:
Type: Nested N × 3 CBOR array
Key: 3
Description: This array contains credentials already registered with the user. If the authenticator detects any of these credentials as existing on the device, it must return an error to prevent the creation of duplicates. All values of the array are encrypted with the GCM key.

∗ **CREDENTIAL TYPE**: *Type*: Integer
Length: 1 Byte
Description: Specifies the type of credential. Currently, only one type is supported by Webauthn, but the design is structured to allow for future expansions
0: PUBL…

∗ **CREDEN…**
Type: Byte…
Length: Va…
Description…

∗ **TRANSP…**
Type: Integ…
Length: 1 B…
Description…
maps to:
0: USB
1: NFC
2: BLE
3: INTE…

– **ATTESTATI…**
Type: Integer
Key: 4
Length: 1 Byte
Description: Sp…
This values is c…
0: NONE
1: INDIREC…
2: DIRECT
3: ENTERP…

– **EXTENSION…**
Type: Nested M…
Key: 5

A.3. Pre-Registration Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 3

…ssage, indicating a Pre-Registration Response.

…ed with the end user's account.

…gistration Request

…SAGE TYPE:
…Integer
…: 5

…ond…

…pose…
…ypte…

…s to impose regarding authe…
…to:

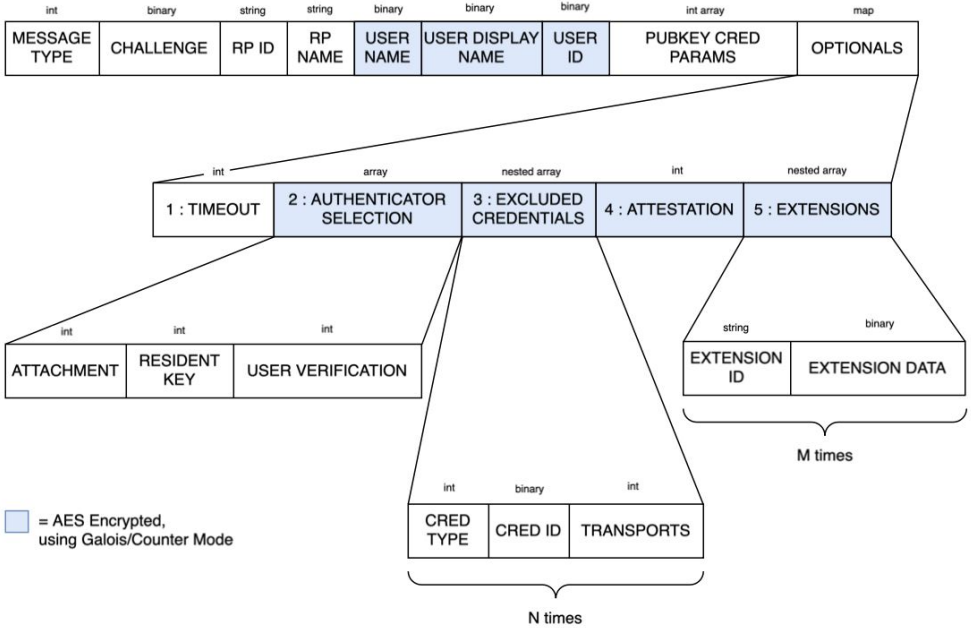
…s to impose regarding disc…
…to:

…to impose regarding user ve…

A.6. Registration Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 6
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Registration Response.
- **ATTESTATION OBJECT**:
  Type: Byte Array
  Length: Variable, up to 2^{13} Bytes
  Description: A binary representation of the attestation object, which contains cryptographic proof of the newly created credential. This object is used by the RP to verify the integrity and origin of the new public key credential. The attestation object is defined by Webauthn, as shown in Figure 17
- **CLIENTDATA JSON**:
  Type: UTF-8 String
  Length: Variable, up to 2^{11} Bytes
  Description: A JSON-serialized string containing the client-side data used during the credential creation process. This includes type, challenge and origin.

int	binary	string
MESSAGE TYPE	ATTESTATION OBJECT	CLIENTDATA JSON



FID

M

Tr

0: REQUIRED
1: PREFERRED
2: DISCOURAGED

A.3. Pre-Registration Response

A.2.

ATTESTATION OBJECT

"fmt": "packed"	"attStmt": ...	"aut

AUTHENTICATOR DATA

32 bytes	1 byte	4 bytes (big-endian uint32)	variable length
RP ID hash	FLAGS	COUNTER	ATTESTED CRED. DATA

ED	AT	0	0	0	UV	0	UP
7							0

AAGUID	L	CREDENTIAL ID	CREDEN
16 bytes	2 bytes	LENGTH L (variable length)	variable

ATTESTATION STATEMENT (in "packed" attestation statement format)

If Basic or Privacy CA:	"alg": ...	"sig": ...	"x5c": ...
If ECDAA:	"alg": ...	"sig": ...	"ecdaaKeyId": ...

[Other attestation statement formats are as defined in their respective sections below]

A.6. Registr

- **MESSAG**
  Type: Inte
  Value: 6
  Length: 1
  Descriptio

- **ATTEST**
  Type: Byte Array
  Length: Variable, up to 2^{13} Bytes
  Description: A binary representation of the attestation object, which contains cryptographic proof of the newly created credential. This object is used by the RP to verify the integrity and origin of the new public key credential. The attestation object is defined by Webauthn, as shown in Figure 17

- **CLIENTDATA JSON**:
  Type: UTF-8 String
  Length: Variable, up to 2^{11} Bytes
  Description: A JSON-serialized string containing the client-side data used during the credential creation process. This includes type, challenge and origin.

int	binary	string
MESSAGE TYPE	ATTESTATION OBJECT	CLIENTDATA JSON

s to impose regardin
to:

s to impose regardi
to:

to impose regarding user ve

A.8. Pre-Authentication Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 8
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Authentication Request.

- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference that links this handshake to the following one.

- **GCM KEY**:
  Type: Byte String
  Length: 28 – 44 Bytes
  Description: A key used for symmetric encryption, utilizing the AES algorithm in conjunction with GCM.

int	binary	binary
MESSAGE TYPE	EPHEMERAL USER ID	GCM KEY

on Response.

map

ONALS

ONS

binary

NSION DATA

= AES Encrypted, using Galois/Counter Mode

CRED TYPE	CRED ID	TRANSPORTS

N times

A.9. Pre-Authentication Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 9
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Authentication Response.
- **USER NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The username associated with the end user's account.

int	string
MESSAGE TYPE	USER NAME

A.8. Pre-Authentication Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 8
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Authentication Request.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference that links this handshake to the following one.
- **GCM KEY**:
  Type: Byte String
  Length: $28 - 44$ Bytes
  Description: A key used for symmetric encryption, utilizing the AES algorithm in conjunction with GCM.

int	binary	binary
MESSAGE TYPE	EPHEMERAL USER ID	GCM KEY

= AES Encrypted, using Galois/Counter Mode

CRED TYPE	CRED ID	TRANSPORTS

N times

0: REQUIRED
1: PREFERRED
2: DISCOURAGED

A.3. Pre-Registration Response

on Response.

map

ONALS

ONS

binary

NSION DATA

If ECDAA:	"alg": ...	"sig": ...	"ecdaaKeyId": ...

[Other attestation statement formats are as defined in their respective sections below]

- **ATTEST**
  Type: Byte Array
  Length: Variable, up to 2^{13} Bytes
  Description: A binary representation of the attestation object, which contains cryptographic proof of the newly created credential. This object is used by the RP to verify the integrity and origin of the new public key credential. The attestation object is defined by Webauthn, as shown in Figure 17
- **CLIENTDATA JSON**:
  Type: UTF-8 String
  Length: Variable, up to 2^{11} Bytes
  Description: A JSON-serialized string containing the client-side data used during the credential creation process. This includes type, challenge and origin.

int	binary	string
MESSAGE TYPE	ATTESTATION OBJECT	CLIENTDATA JSON

0: REQUIRED
1: PREFERRED
2: DISCOURAGED

A.9. Pre-Authentication Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 9
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Authentication Response.
- **USER NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The username associated with the end user's account.

A.8. Pre-Authentication Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 8
  Length: 1 Byte
  Description: Specifies the type of message, indicating a Pre-Authentication Request.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference that links this handshake to the following one.

A.10. Authentication Indication

- **MESSAGE TYPE**:
  Type: Integer
  Value: 10
  Length: 1 Byte
  Description: Specifies the type of message, indicating an Authentication Indication.
- **EPHEMERAL USER ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: A ephemeral reference to a previous handshake.

int	binary
MESSAGE TYPE	EPHEMERAL USER ID

A.9. Pre-Authentication Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 9
  Length: 1 Byte
  Description: Specifies the type of message, indicating
  sponse.
- **USER NAME**:
  Type: UTF-8 String
  Length: Variable, up to 255 Bytes
  Description: The username associated with the end us

A.10. Authenti

- **MESSAGE**
  Type: Integer
  Value: 10
  Length: 1 Byt
  Description: S
- **EPHEMER**
  Type: Byte St
  Length: Varia
  Description:

int	binary
MESS TYP	EPHEMERAL USER ID

Value: 6
Length: 1
Descriptio

If ECDAA:

[Other attestatior

- **ATTEST**
  Type: Byte Array
  Length: Variable, up to 2^{13} Bytes
  Description: A binary represent
  cryptographic proof of the newly
  to verify the integrity and origin
  object is defined by Webauthn, a
- **CLIENTDATA JSON**:
  Type: UTF-8 String
  Length: Variable, up to 2^{11} Bytes
  Description: A JSON-serialized s
  the credential creation process. T

int		
MESSAGE TYPE	ATTESTATION OBJECT	CLIENTDATA JSON

A.11. Authentication Request

- **MESSAGE TYPE**:
  Type: Integer
  Value: 11
  Length: 1 Byte
  Description: Specifies the type of message, indicating an Authentication Request.
- **CHALLENGE**:
  Type: Byte String
  Length: 16 – 64 Bytes
  Description: Random challenge from the RP.
- **OPTIONALS**:
  Type: CBOR map
  Description: Contains optional values that can be probed by their keys.
  - **TIMEOUT**:
    Type: Integer
    Key: 1
    Length: 4 Bytes
    Description: Maximum time, in milliseconds, that the client should wait for the user to complete the authentication process.
  - **RP ID**:
    Type: UTF-8 String
    Key: 2
    Length: Variable, up to 256 Bytes
    Description: Relying Party identifier.
  - **USER VERIFICATION**:
    Type: Integer
    Key: 3
    Length: 1 Byte
    Description: Specifies the level of user verification required for authentication. Enum value that maps to:
    0: REQUIRED
    1: PREFERRED
    2: DISCOURAGED
  - **ALLOW CREDENTIALS**:
    Type: Nested N × 3 CBOR array
    Key: 4
    Description: List of credentials permitted for use in the authentication.

int	binary
MESSAGE TYPE	EPHEMERAL USER ID

st

essage, indi

that links

tic encryp

ERAL
ID

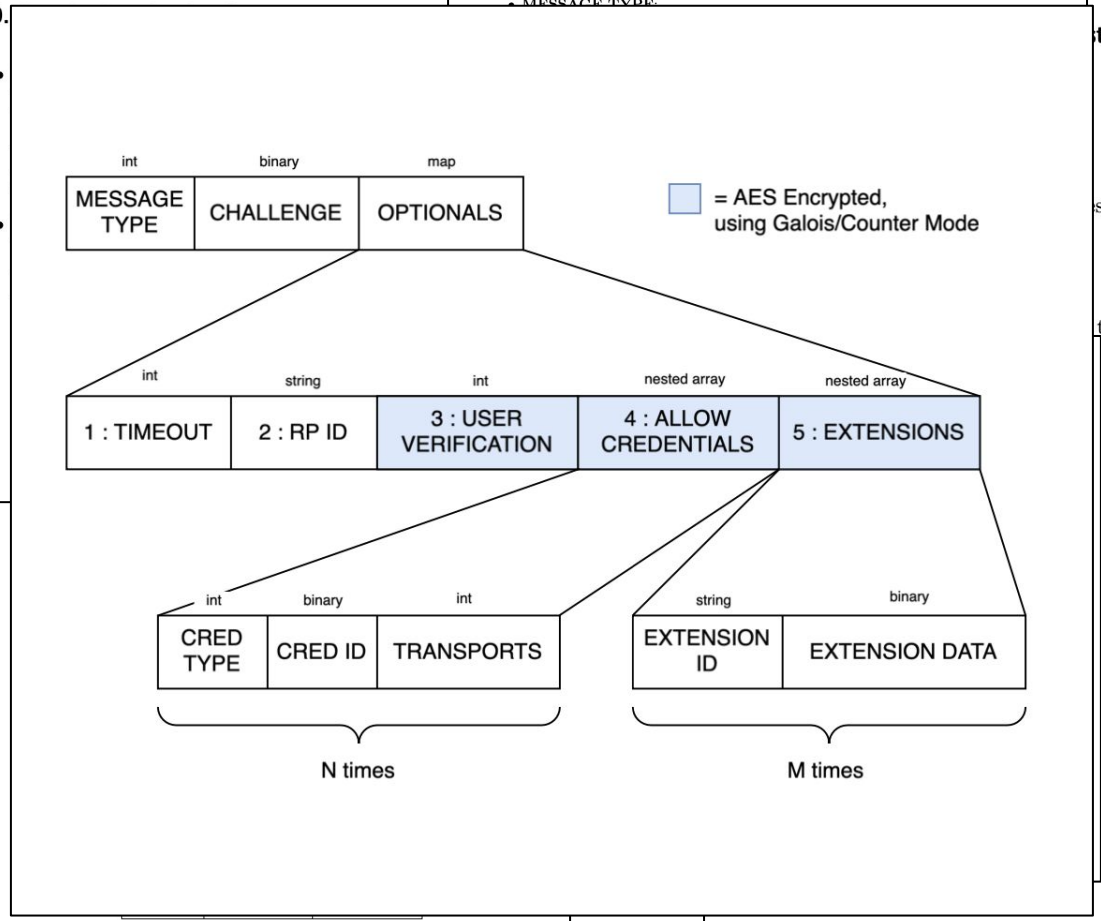
A.3. Pre-Registration Response

- **CREDENTIAL TYPE**:
  Type: Integer
  Length: 1 Byte
  Description: Specifies the type of credential. Currently, only one type is supported by Webauthn, but the design is structured to allow for future expansions. Enum value that maps to:
  0: PUBLIC_KEY
- **CREDENTIAL ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: Unique identifier for the credential.
- **TRANSPORTS**:
  Type: Integer
  Length: 1 Byte
  Description: Mode of transportation for this credential. Enum value that maps to:
  0: USB
  1: NFC
  2: BLE
  3: INTERNAL

– **EXTENSIONS**:
Type: Nested M × 2 CBOR array
Key: 5
Description: Custom extension data. All values of the array are encrypted with the GCM key.

- **EXTENSION ID**:
  Type: UTF-8 String
  Length: Variable, up to 256 Bytes
  Description: Unique identifier of the extension.
- **EXTENSION DATA**:
  Type: Byte String
  Length: Variable, up to 2^{12} Bytes
  Description: Data corresponding to the extension, which may include any necessary parameters or configuration settings to support the extension's function.

N times



A.11. Authentication Request

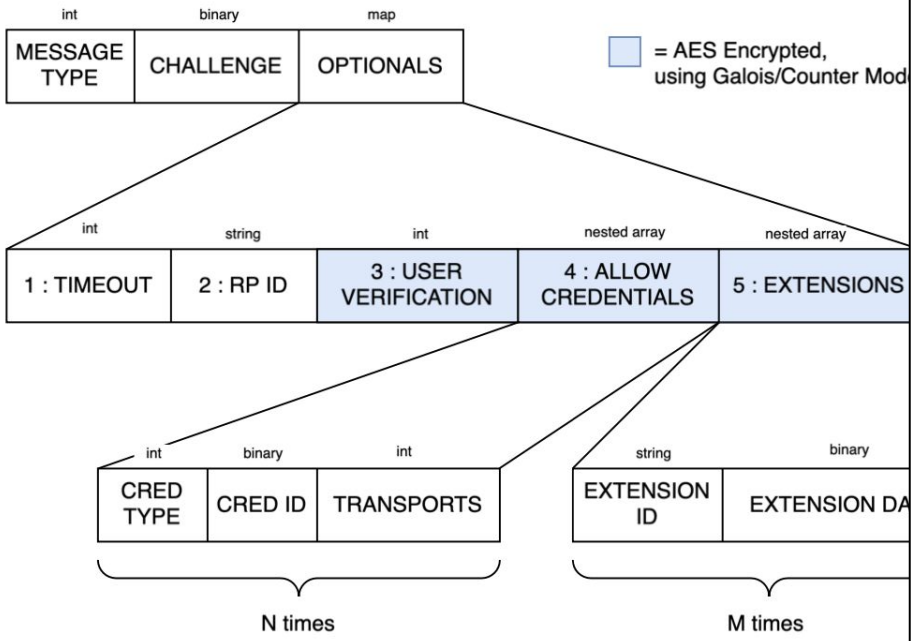
A.3. Pre-Registration Response

- ∗ **CREDENTIAL TYPE**:
  Type: Integer
  Length: 1 Byte
  Description: Specifies the type of credential. Currently, only one type is supported by Webauthn, but the design is structured to allow for future expansions. Enum value that maps to:
  0: PUBLIC_KEY
- ∗ **CREDENTIAL ID**:
  Type: Byte String
  Length: Variable, up to 256 Bytes
  Description: Unique identifier for the credential.
- ∗ **TRANSPORTS**:
  Type: Integer
  Length: 1 Byte
  Description: Mode of transportation for this credential. Enum value that maps to:
  0: USB
  1: NFC
  2: BLE
  3: INTERNAL

– **EXTENSIONS**:
Type: Nested M × 2 CBOR array
Key: 5
Description: Custom extension data. All values of the array are encrypted with the GCM key.

- ∗ **EXTENSION ID**:
  Type: UTF-8 String
  Length: Variable, up to 256 Bytes
  Description: Unique identifier of the extension.
- ∗ **EXTENSION DATA**:
  Type: Byte String
  Length: Variable, up to 2^{12} Bytes
  Description: Data corresponding to the extension, which may include any necessary parameters or configuration settings to support the extension's function.

N times



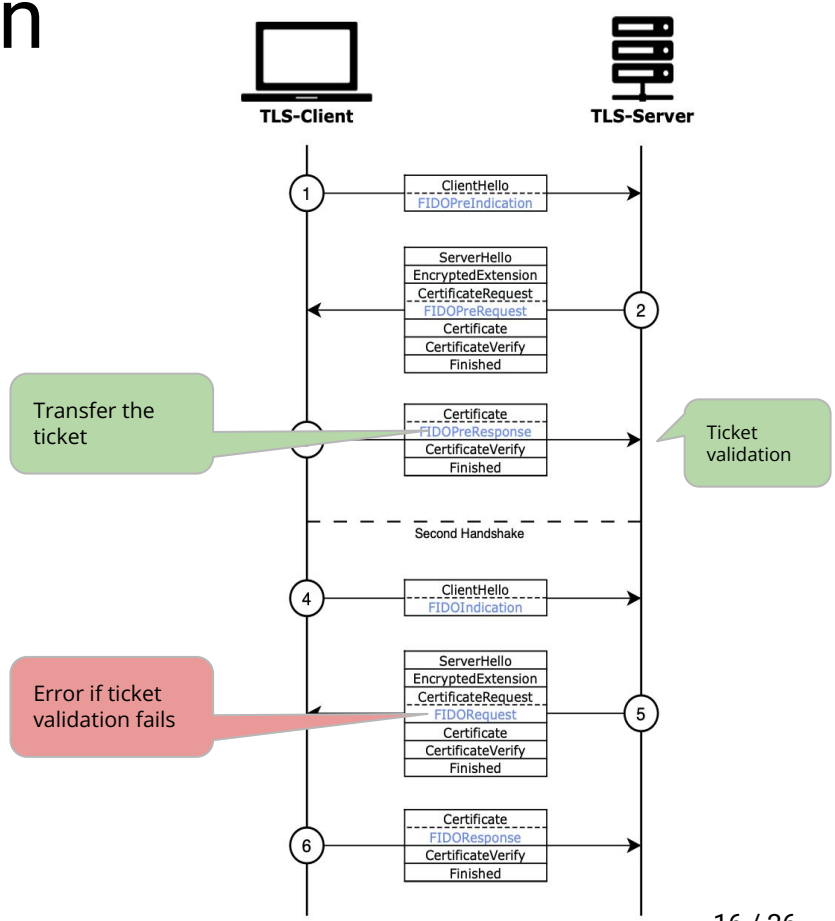
A.12. Authentication Response

- **MESSAGE TYPE**:
  Type: Integer
  Value: 12
  Length: 1 Byte
  Description: Specifies the type of message, indicating an Authentication Response.
- **CLIENTDATA JSON**:
  Type: UTF-8 String
  Length: Variable, up to 2^{11} Bytes
  Description: A JSON-serialized string containing the client-side data used during the authentication process. This includes type, challenge, and origin.
- **AUTHENTICATOR DATA**:
  Type: Byte Array
  Length: Variable, up to 512 Bytes
  Description: A binary representation of the authenticator data, which contains data from the authenticator, including flags and counters.
- **SIGNATURE**:
  Type: Byte Array
  Length: Variable, up to 256 Bytes
  Description: A digital signature over the concatenation of the authenticator data and the clientDataHash.
- **OPTIONALS**:
  Type: CBOR map
  Description: Contains optional values that can be probed by their keys.
  - **USER HANDLE**:
    Type: Byte String
    Key: 1
    Length: Variable, up to 64 Bytes
    Description: The user ID, that uniquely identifies the user. If discoverable credentials were used, this field is mandatory.
  - **SELECTED CREDENTIAL ID**:
    Type: Byte String
    Key: 2
    Length: Variable, up to 256 Bytes
    Description: The identifier of the credential that was used in the authentication.
  - **CLIENT EXTENSION OUTPUT**:
    Type: Nested N × 2 CBOR array
    Key: 3
    Description: Custom extension data returned by the client.

FIDO2 as TLS 1.3 Extension

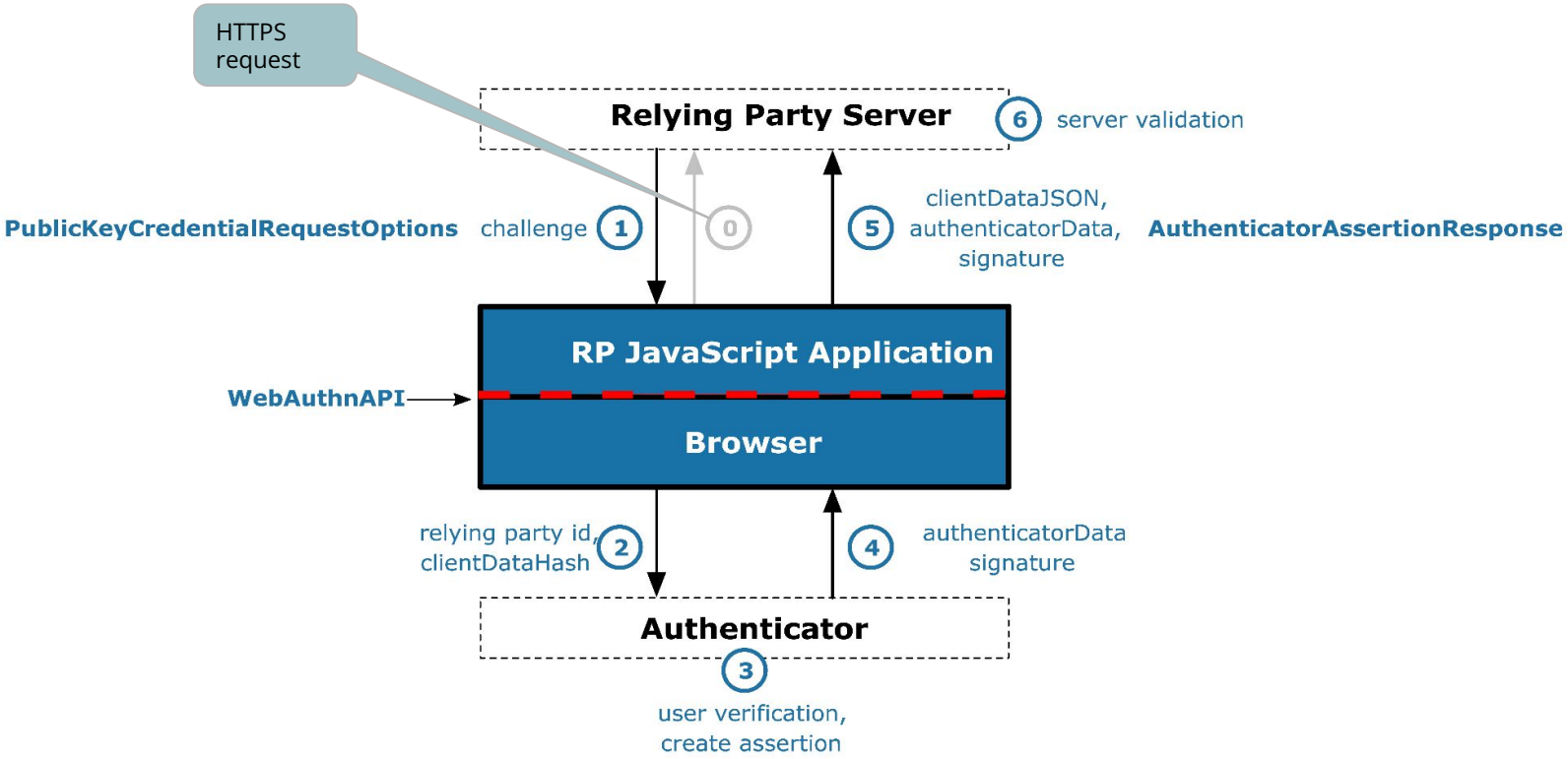
Control Mechanism for Key Registration

- Problem: Not every user is allowed to register a key
  - In traditional FIDO, a preliminary authentication step (outer web context) gatekeeps new registrations
- Solution: Shared secret (**ticket**) distributed to both peers out of band
- Semantic meaning of tickets (not part of the implementation):
  - User name + ID
  - Expiration Date
  - Issuer Information
  - Purpose + Usage Context
  - Authenticator Properties



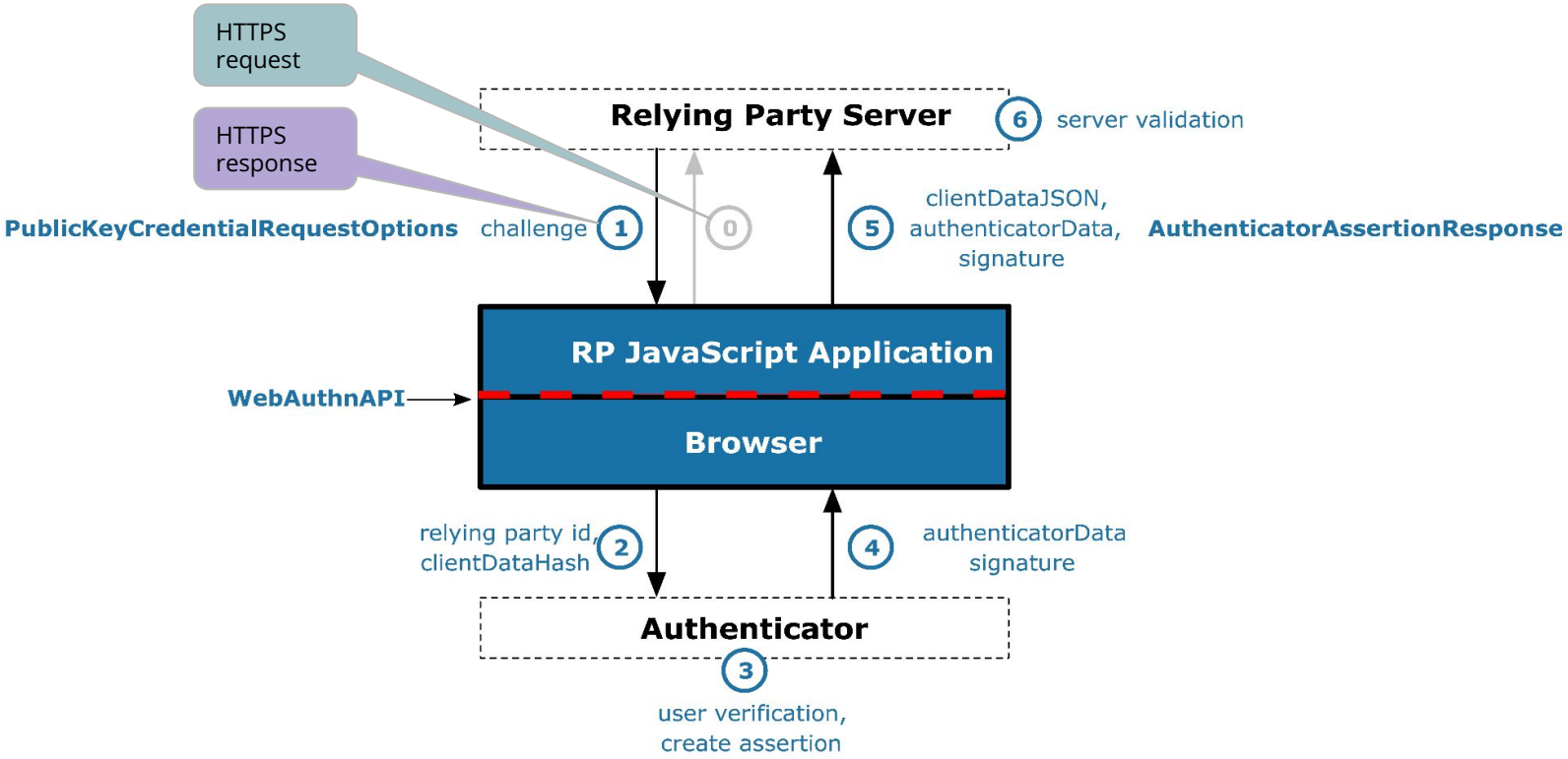
FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer



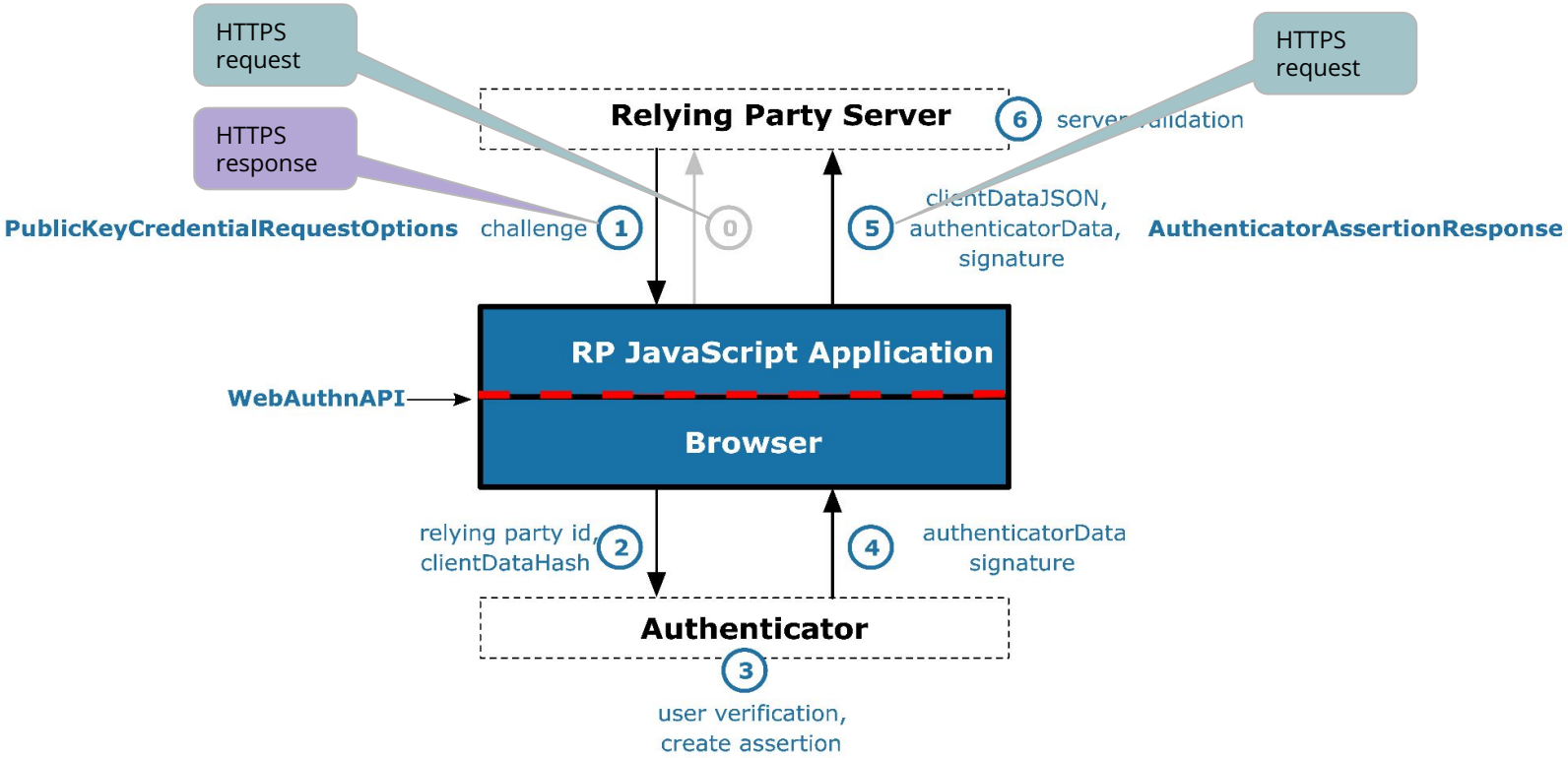
FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer



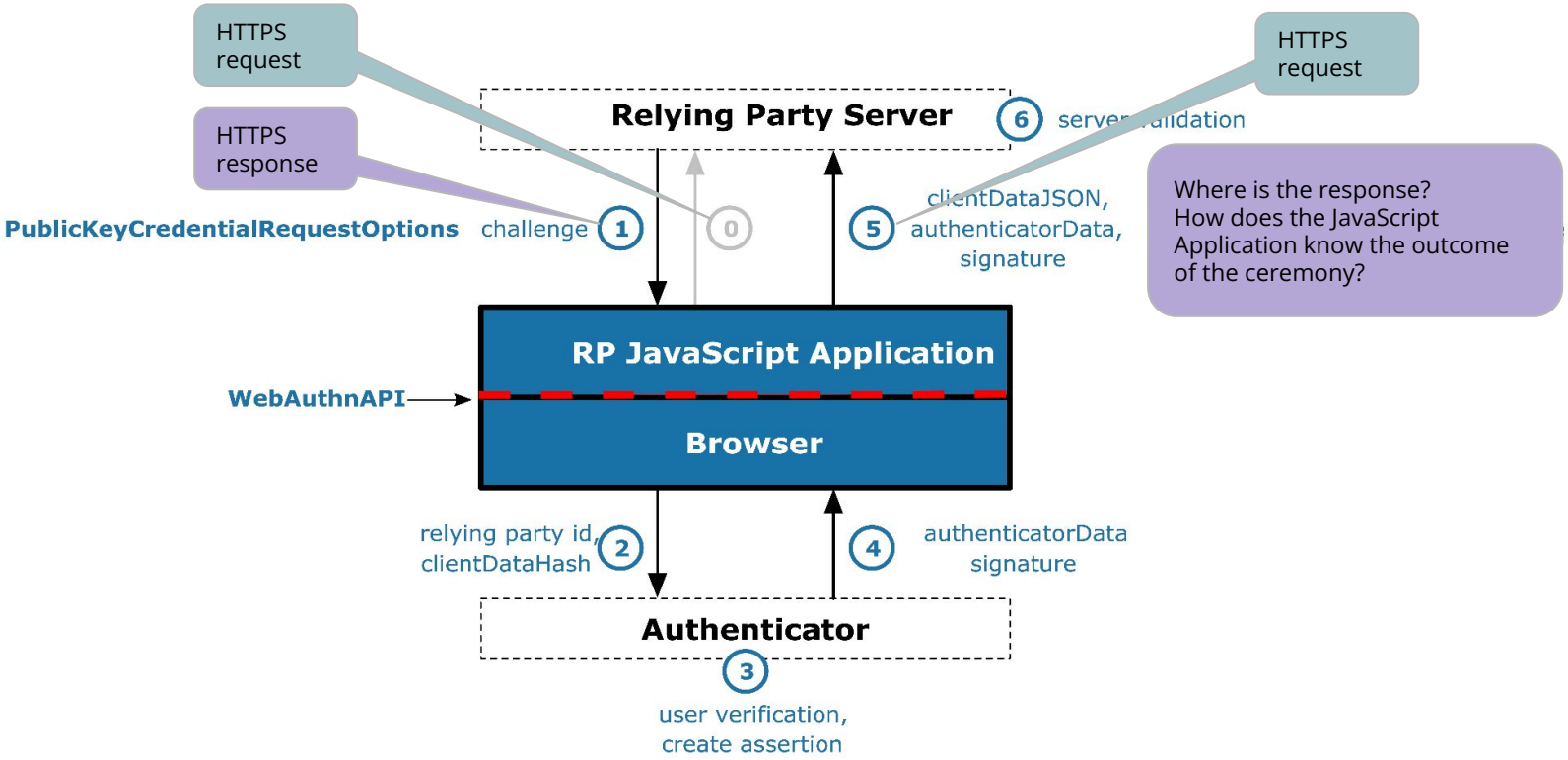
FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer



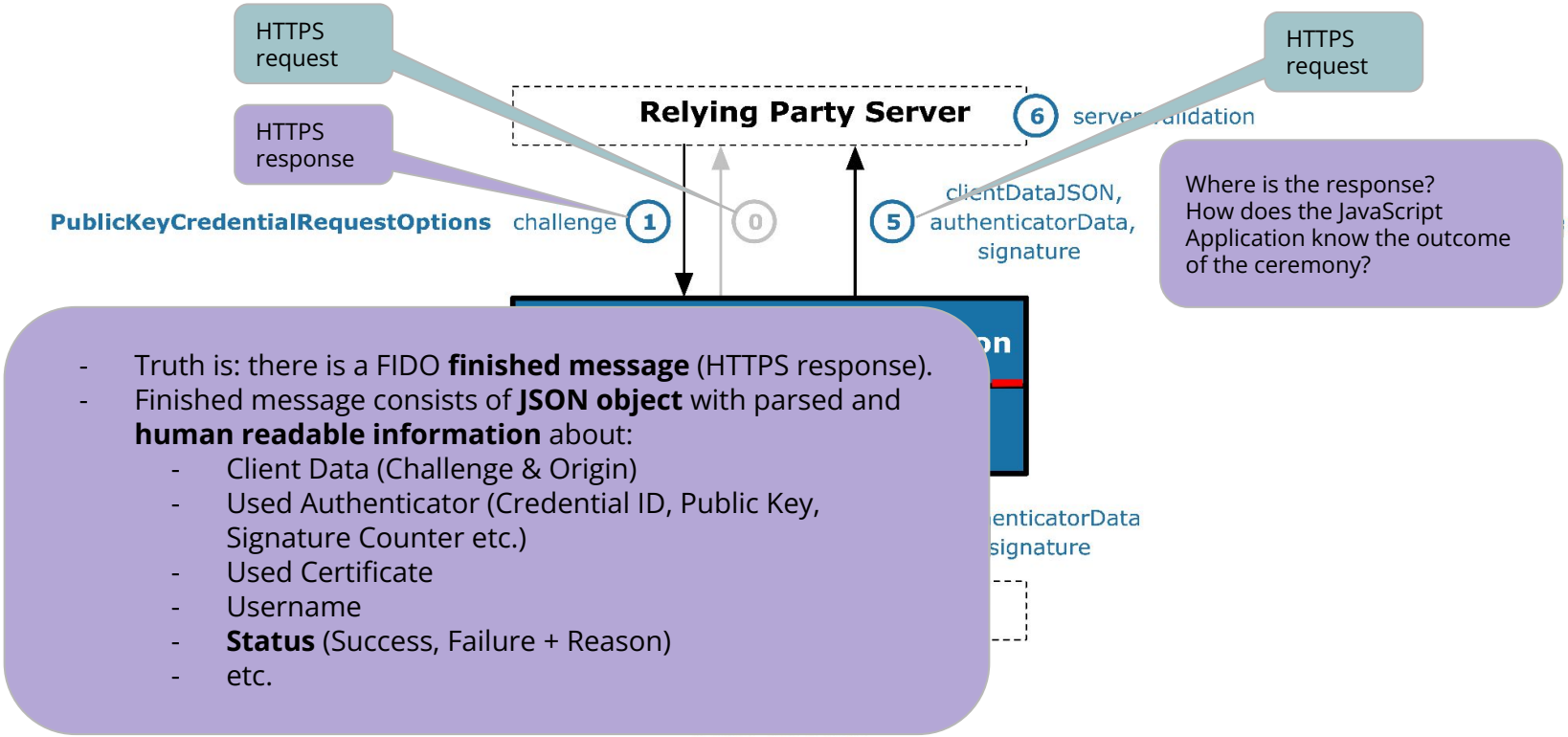
FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer



FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer



FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer

How to send the finished message **back to the client**?

FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer

How to send the finished message **back to the client**?

- Observation:
  - Only the **status** is new information to the client
  - All the other data is **redundant** because client already relayed this information before from the authenticator
    - Not necessarily human readable
    - Client must parse CBOR blobs

FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer

How to send the finished message **back to the client**?

- Observation:
  - Only the **status** is new information to the client
  - All the other data is **redundant** because client already relayed this information before from the authenticator
    - Not necessarily human readable
    - Client must parse CBOR blobs

- Solution: Simple status codes can be communicated using **TLS-Alerts** by
  - Extending the TLS-Alerts: Patching the TLS library (long process until standardisation)
  - Reusing TLS-Alerts: Map existing alerts to new FIDO status codes (Problem: **ambiguity**)

FIDO2 as TLS 1.3 Extension

TLS Alerts & Communication to the Application Layer

How to

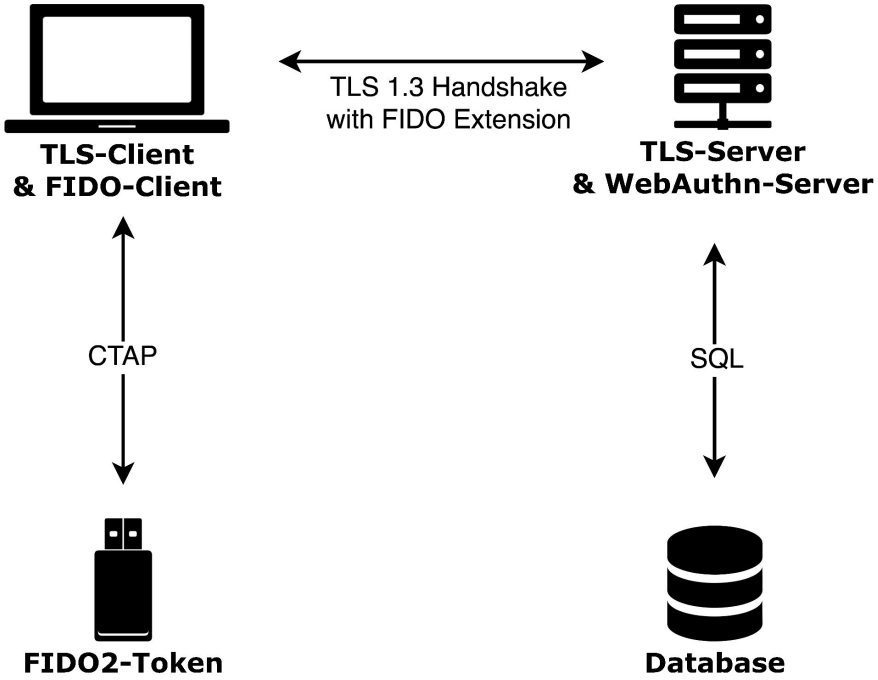
- Ob
  -
  - ore
- So by
  - n)
  - ity)

- **Success**: No alert is necessary; normal operation continuation.
- **Invalid Credentials**: Mapped to `access_denied (49)`, indicating that the provided credentials do not grant access.
- **Device Incompatibility**: Mapped to `illegal_parameter (47)`, repurposed to indicate unsupported device characteristics.
- **Timeout**: Mapped to `user_canceled (90)`, this alert signifies that the operation ended due to a timeout. This could be because data processing took too long or required user action wasn't completed promptly.
- **Internal Error**: Mapped to `internal_error (80)`, used for reporting unspecified system failures.

Implementation

fidoSSL

- TLS library: OpenSSL Cryptography and SSL/TLS Toolkit
  - Written in C
  - Used Version: v3.2.1 (latest at the time of writing)
- Dependencies:
  - ***libfido2***: Implements CTAP
  - ***tinycbor***: CBOR library know for it's small footprint
  - ***sqlite3***: Persistent database for the WebAuthn-Server
  - ***libjansson***: Construct and parse JSON objects.



Implementation

fidoSSL

- Features:
  - FIDO key registration (double handshake)
  - FIDO authentication with discoverable credentials (single handshake)
  - Uses a shared secret (**ticket**) to gatekeep key registrations
- Limitations
  - No authentication with non-discoverable credentials
  - Semantic meaning of tickets is not evaluated
  - Attestation not implemented
  - No support for FIDO extensions
  - No support for platform authenticators (only USB tokens)
  - Only supported signing algorithm: ES256 (ECDSA with SHA256 on a P-256 curve)
- Usage: Wait for Live Demo

Implementation

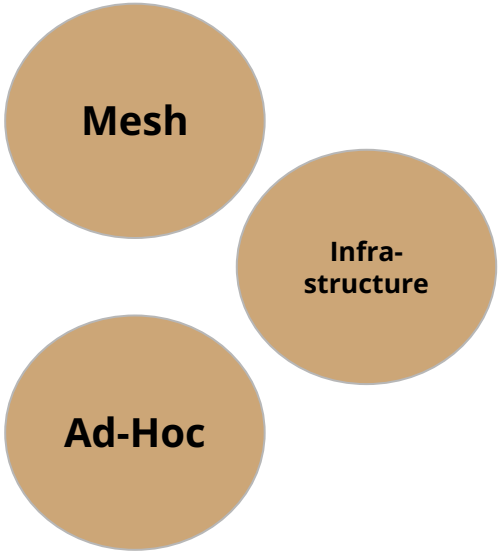
hostap-fido2

Goal: Showcase the practical use of the FIDO2 TLS1.3 Extension in **Wireless Networks**

Implementation

hostap-fido2

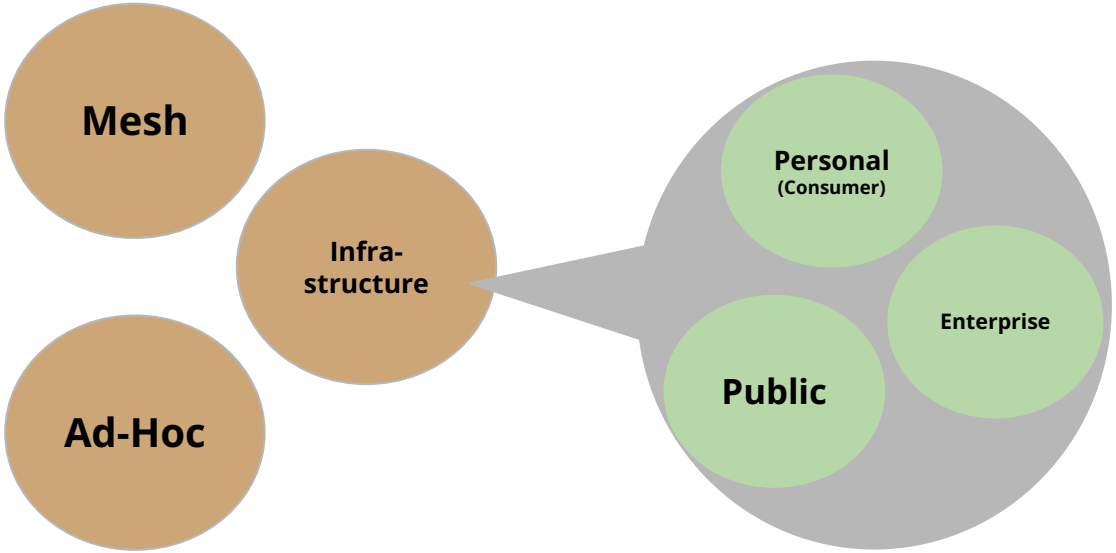
Goal: Showcase the practical use of the FIDO2 TLS1.3 Extension in **Wireless Networks**



Implementation

hostap-fido2

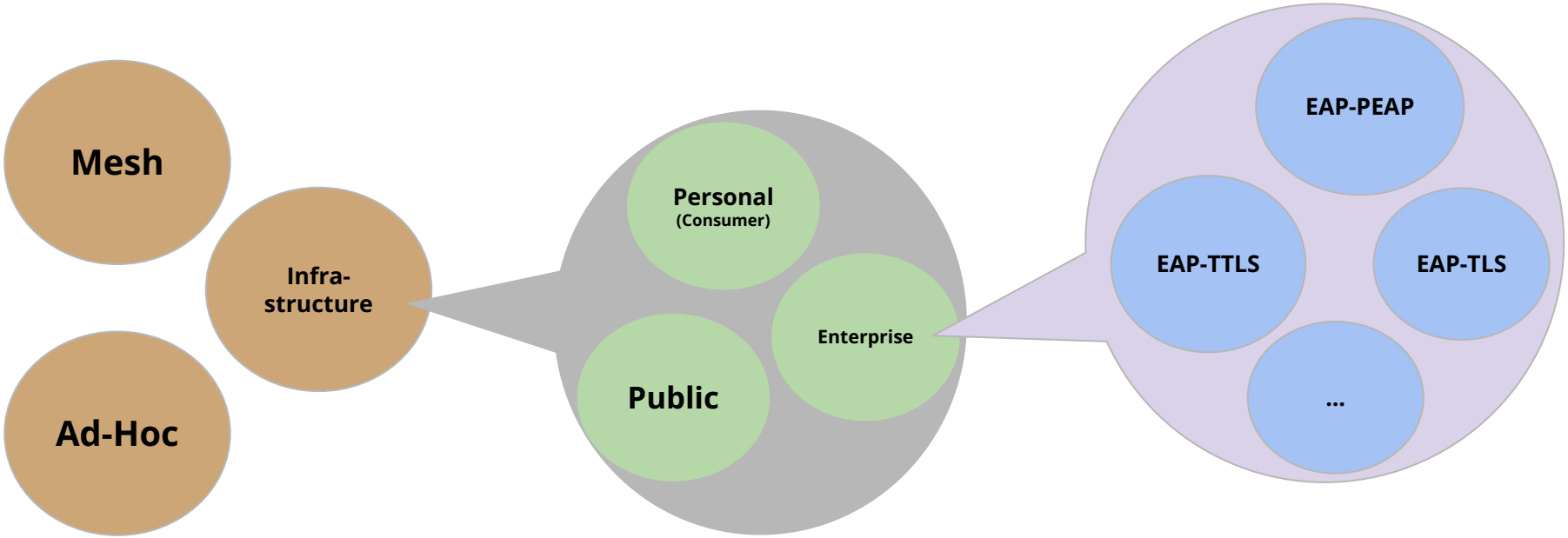
Goal: Showcase the practical use of the FIDO2 TLS1.3 Extension in **Wireless Networks**



Implementation

hostap-fido2

Goal: Showcase the practical use of the FIDO2 TLS1.3 Extension in **Wireless Networks**



Implementation

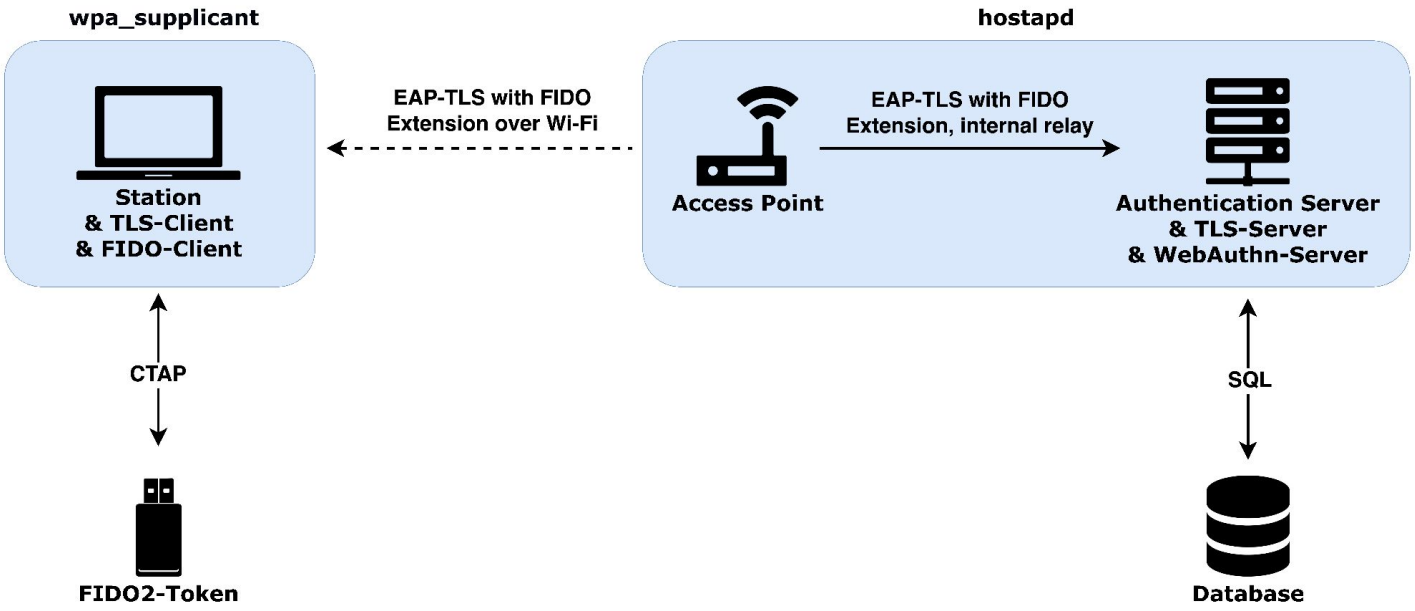
hostap-fido2

The *hostap* project packages:

- ***hostapd***: A daemon used to turn a network adapter into a **wireless access point**, enabling a device to broadcast **Wi-Fi**
  - Manages authentication, encryption, access control, QoS etc.
- ***wpa_supplicant:*** The counterpiece for *hostapd*, running on the client (supplicant)

Implementation

hostap-fido2 (Topology)



Implementation

hostap-fido2

Scope:

- FIDO authentication with USB token instead of client certificates
- Key registration out of band (e.g. through HU-Website with eduroam)
- Quick & dirty implementation:
  - Server config is propagated to *hostapd.conf*
  - Client config is hardcoded

Live Demo and Q&A

Acknowledgments 🙏

I would like to thank:

- **Dr. Wolf Müller**, my supervisor, for his excellent support with all my questions and problems.
- **Prof. Dr. Jens Peter Redlich**, my primary examiner, for evaluating my work.
- **Prof. Dr. Joel Rybicki**, my co-examiner, for his contributions.