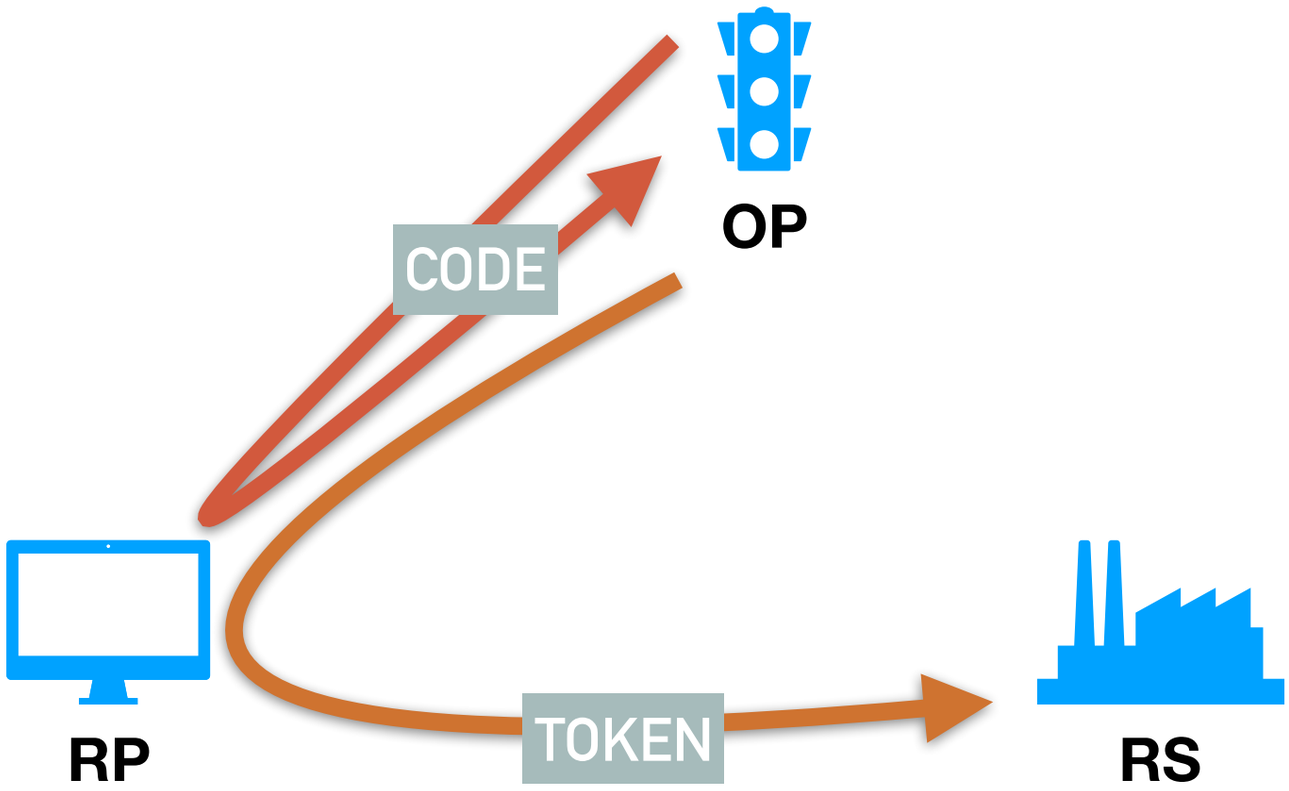
EXTENSION RFCS

PUBLISHED RFCS – BASIC

- OAuth 2.0 Threat Model and Security Considerations (RFC6819)
- JSON Web Token (JWT) (RFC 7519)

OPEN ISSUES

- The meaning of a token ?
- Who am I talking to ?
- Who can use a token ?



PUBLISHED RFCS – CLIENT AUTHENTICATION & AUTHORIZATION

- Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants (RFC 7521)

- Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants (RFC 7522)

- JSON Web Token (JWT) Profile For OAuth 2.0 Client Authentication and Authorization Grants (RFC 7523)

RFC7521 -ASSERTION FRAMEWORK FOR OAUTH 2.0 CLIENT AUTHENTICATION AND AUTHORIZATION GRANTS

- Using Assertions as Authorization Grants
  - grant_type
  - assertion
  - scope

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&
assertion=PHNhbWxwOl...[omitted for brevity]…ZT4
```

- Using Assertions for Client Authentication
  - client_assertion_type
  - client_assertion
  - client_id

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Asaml2-bearer&
client_assertion=PHNhbW...[omitted for brevity]...ZT
```

PUBLISHED RFCS – DYNAMIC CLIENT REGISTRATION

- OAuth 2.0 Dynamic Client Registration Protocol (RFC 7591)
- OAuth 2.0 Dynamic Client Registration Management Protocol (RFC 7592 EXP)

PUBLISHED RFCS – TOKEN HANDLING

- OAuth 2.0 Token Revocation (RFC 7009)
- OAuth 2.0 Token Introspection (RFC 7662)

PUBLISHED RFCS – SECURITY

- Proof Key for Code Exchange by OAuth Public Clients (RFC 7636)
- Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs) (RFC 7800)

PKCE (RFC 7636)

A. The client creates and records a secret named the ”code_verifier" and derives a transformed version "t(code_verifier)" (referred to as the "code_challenge"), which is sent in the OAuth 2.0 Authorization Request along with the transformation method ”t_m".

B. The Authorization Endpoint responds as usual but records "t(code_verifier)" and the transformation method.

C. The client then sends the authorization code in the Access Token Request as usual but includes the "code_verifier" secret generated at (A).

D. The authorization server transforms "code_verifier" and compares it to "t(code_verifier)" from (B). Access is denied if they are not equal.

PKCE - EXAMPLE

Setup:

code_verifier=dBjftJeZ4CVPmB92K27uhbUJU1p1r_wW1gFWFOEjXk
code_challenge_method=S256

code_challenge_method(code_verifier) = code_challenge

code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw

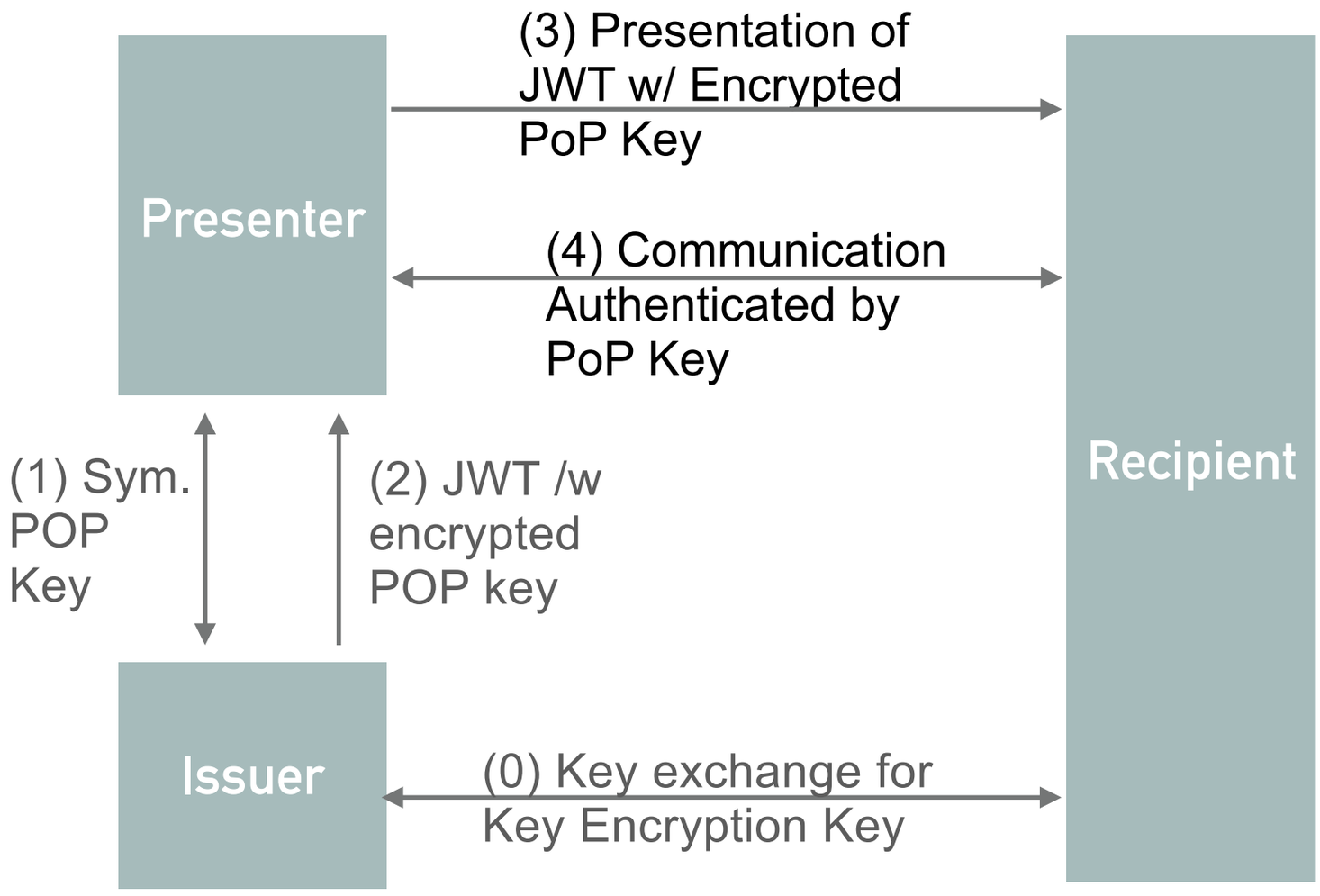
Authorization request:

```
code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM&code_challenge_method=S256
```

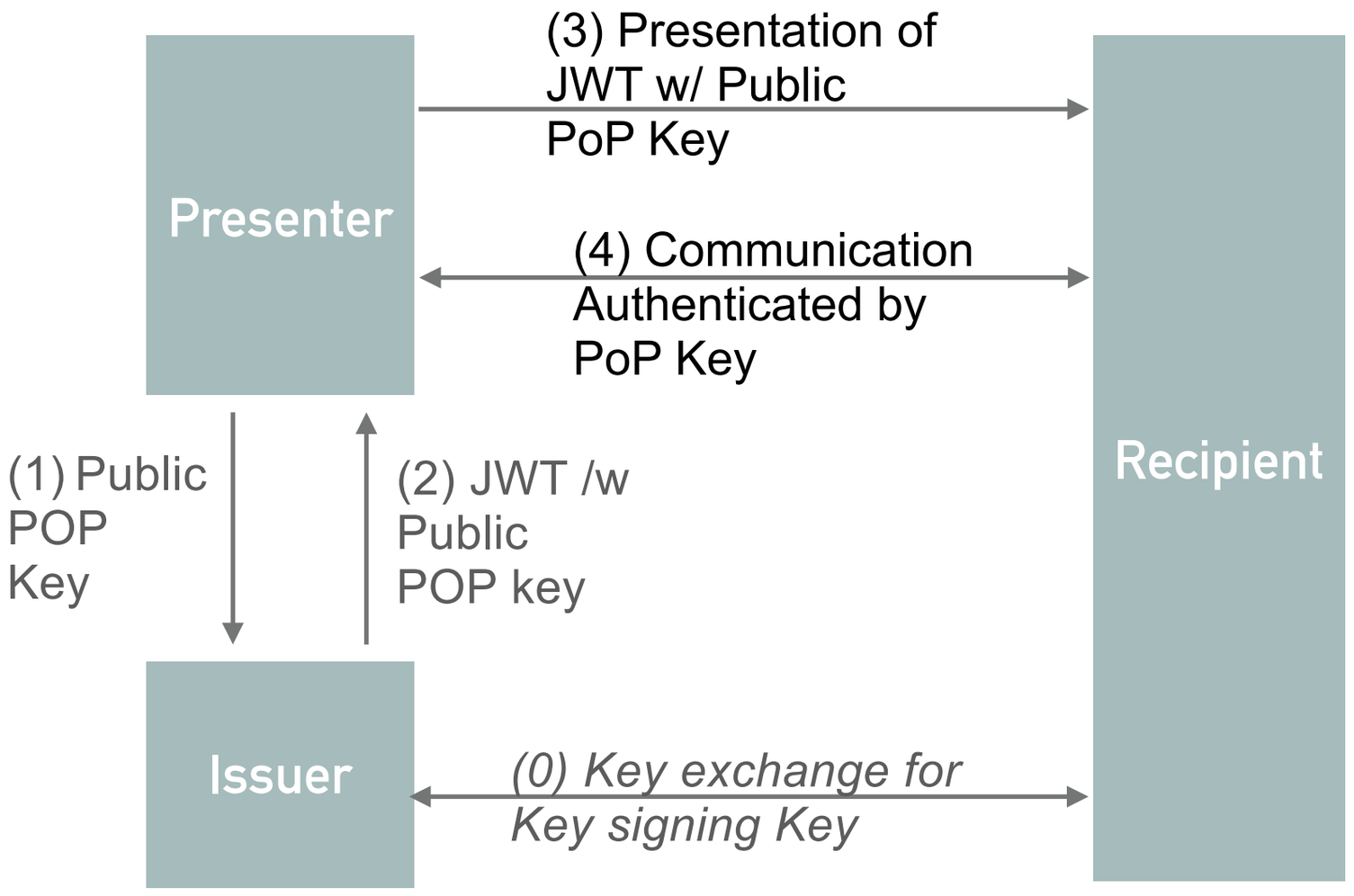
Token request:

```
code_verifier=dBjftJeZ4CVPmB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

RFC7800 PROOF OF POSSESSION WITH SYMMETRIC KEYS



RFC7800 POP WITH ASYMMETRIC KEYS



RFC 8176 AUTHENTICATION METHODS REFERENCES

Each "amr" value provides an identifier for a family of closely related authentication methods

Examples:

otp - One-time password
pwd - Password based authentication
rba - Risk based authentication
sc - Smart card
mfa - Multiple-factor authentication
kba - Knowledge-based authentication

RFC 8252 OAUTH 2.0 FOR NATIVE APPS (BCP)

Abstract:

OAuth 2.0 authorization requests from native apps should only be made through external user-agents, primarily the user's browser. This specification details the security and usability reasons why this is the case and how native apps and authorization servers can implement this best practice.

WORKING GROUP DRAFTS

WORKING GROUP DRAFTS – IESG PROCESSING

- draft-ietf-oauth-discovery
- draft-ietf-oauth-jwsreq

  Request parameters in a JSON Web Token (JWT)

WORKING GROUP DRAFTS – ACTIVE

- draft-ietf-oauth-device-flow

  The device flow is suitable for OAuth 2.0 clients executing on devices that do not have an easy data-entry method

- draft-ietf-oauth-mlts

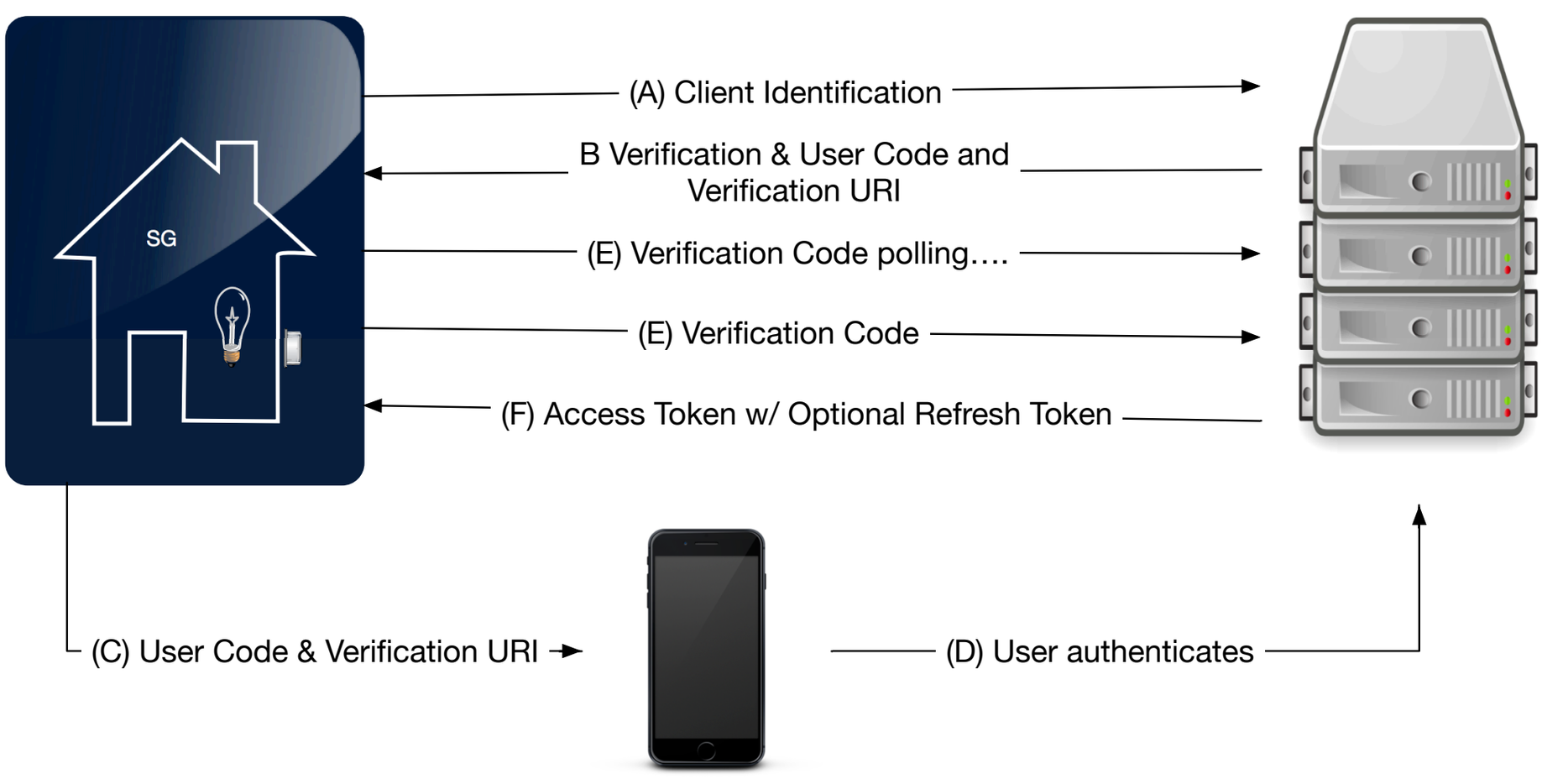
  Mutual TLS profiles for OAuth clients

- draft-ietf-oauth-security-topics
- draft-ietf-oauth-token-binding
- draft-ietf-oauth-token-exchange
- draft-ietf-oauth-jwt-bcp-00

  JSON Web Token Best Current Practices

DRAFT-IETF-OAUTH-DEVICE-FLOW

- OAuth 2.0 clients executing on devices that do not have an easy data-entry method
- end-user has separate access to a user-agent on another computer or device



DRAFT-IETF-OAUTH-TOKEN-BINDING

- Cryptographically binds tokens to the TLS connection between the client and the token endpoint.
- Two use cases
  - First party (Refresh Tokens or Access Tokens)
  - Federation Use Cases