認証と認可と君と

The Triple-A: Authentication, Authorization, and Android

2018/02/09

Obligatory Self Introduction

- @ken5scal
- 2018/0 〜 : **FOLIO**
- Security Architect/Engineer

©FromSoftware

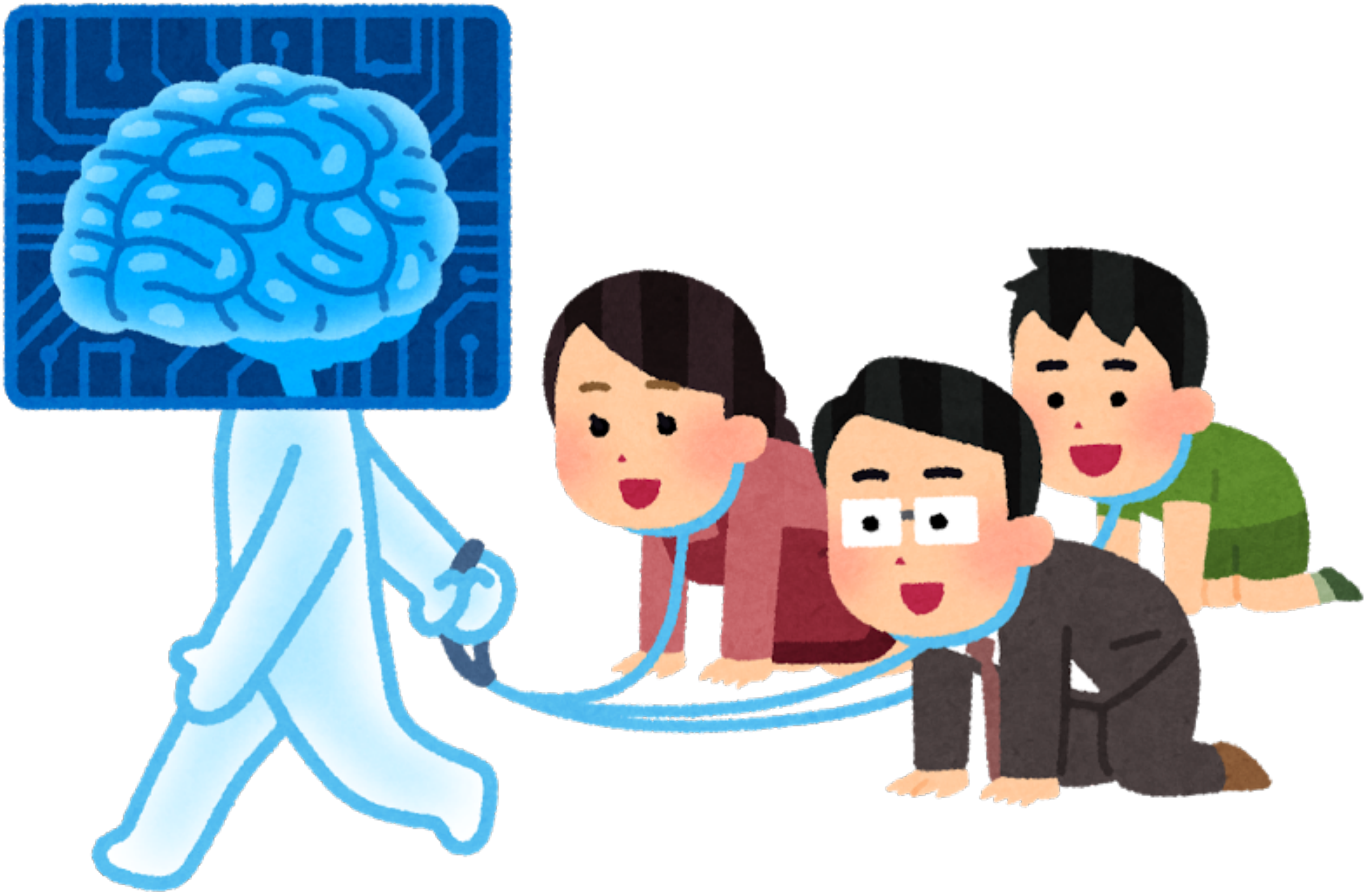
Got FINTECH?

FINTECH

https://www.pymnts.com/news/b2b-payments/2016/treasury-fintech-east-partners-report-investment-corporate-adoption-financial-technology/

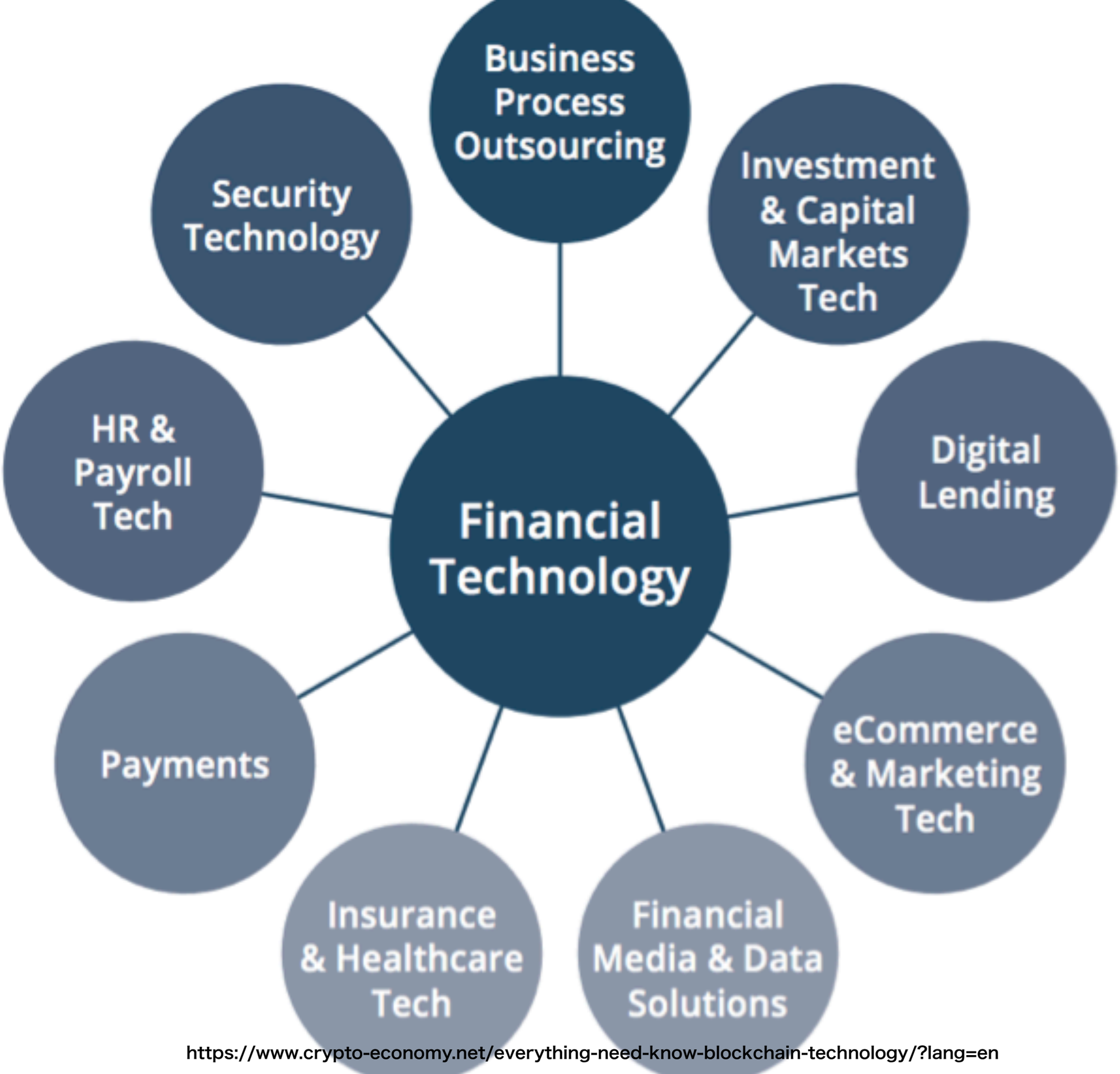
3 main finTECH

https://www.crypto-economy.net/everything-need-know-blockchain-technology/?lang=en



One More Thing…

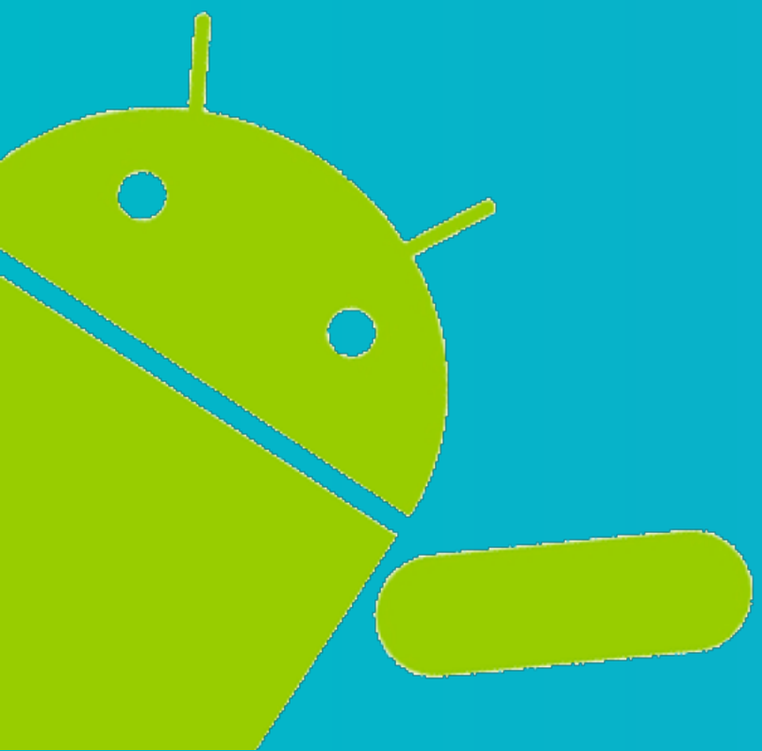
API



https://www.crypto-economy.net/everything-need-know-blockchain-technology/?lang=en

New Value, New Risk

AuthN and AuthZ



in Android

Overview

1. OAuth 2.0 for Native Application
2. FIDO UAF
3. ~~Firebase Authentication~~

Overview

1. OAuth 2.0 for Native Application
2. FIDO UAF
3. ~~Firebase Authentication~~

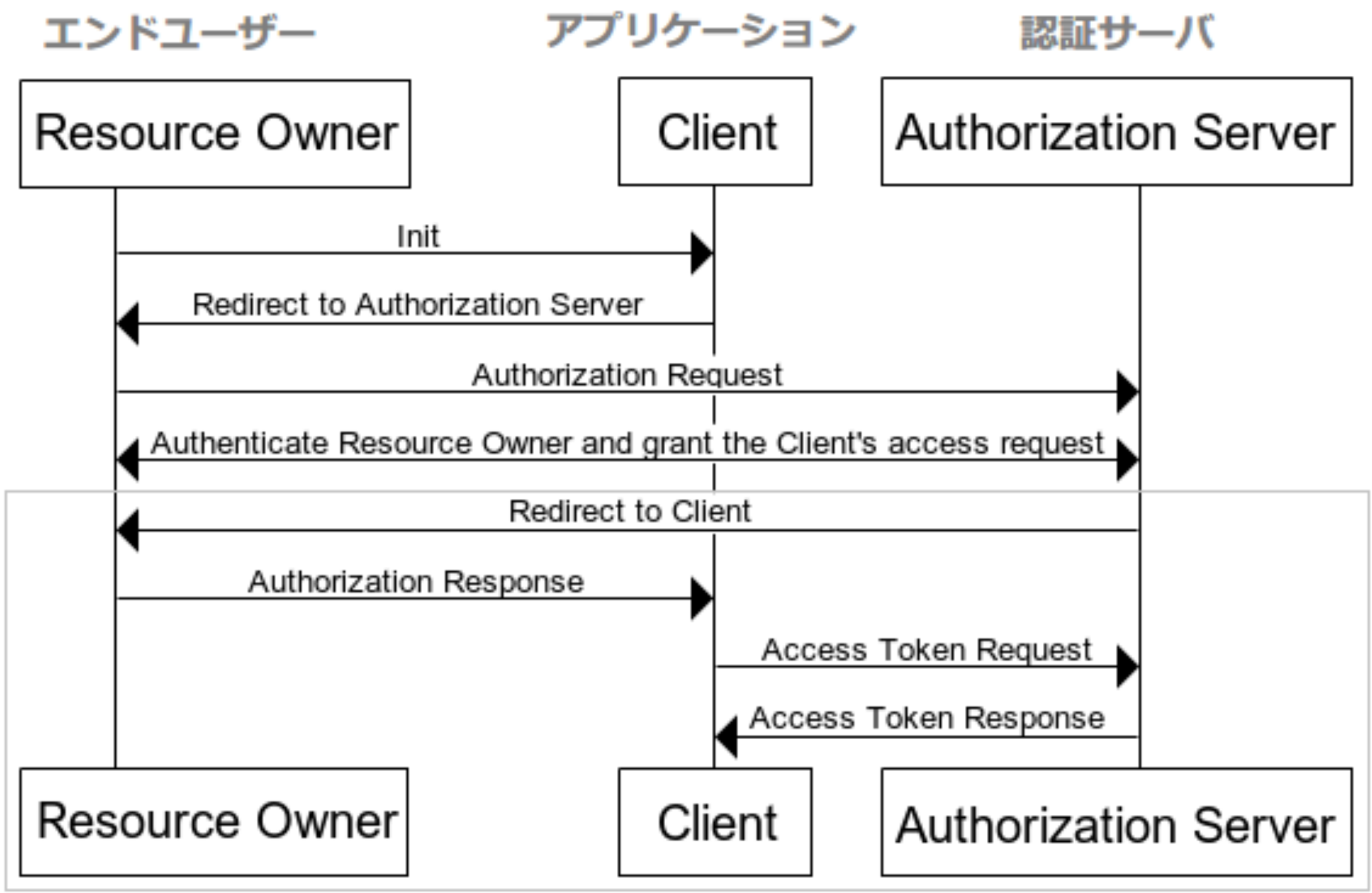
1. Androidモダンプログラミング
2. ken5scal.hatenablog.com

for Android

OAuth 2.0 in nutshell

- OAuth … is **a delegation protocol**. (RFC6749)
- 4 Roles
- **OAuth2.0 for Native Application** (RFC8252)

OAuth as Delegation Protocol



http://www.atmarkit.co.jp/ait/articles/1710/24/news011.html

OAuth 2.0 Main Roles

Client	● **本セッションの主役** ● ROを代理し、保護されたリソースにアクセスするソフト ● AndroidであればAndroidアプリ　もしくはSDK
Resource Owner (RO)	● 保護されたリソースへのアクセス権限を保有する ● 保護されたリソースは基本APIの形を取る事が多い ● 通常、エンドユーザー（人）
Authorization Server (AS)	● アクセストークンをクライアントに発行するサーバー ● 保護されたリソースに信頼されている ● ROの認証と権限付与が完了した後にトークンを発行
Resource Server (RS)	● 保護されたリソースをホスティングするサーバー ● トークンからリソースへのリクエスト範囲を制限

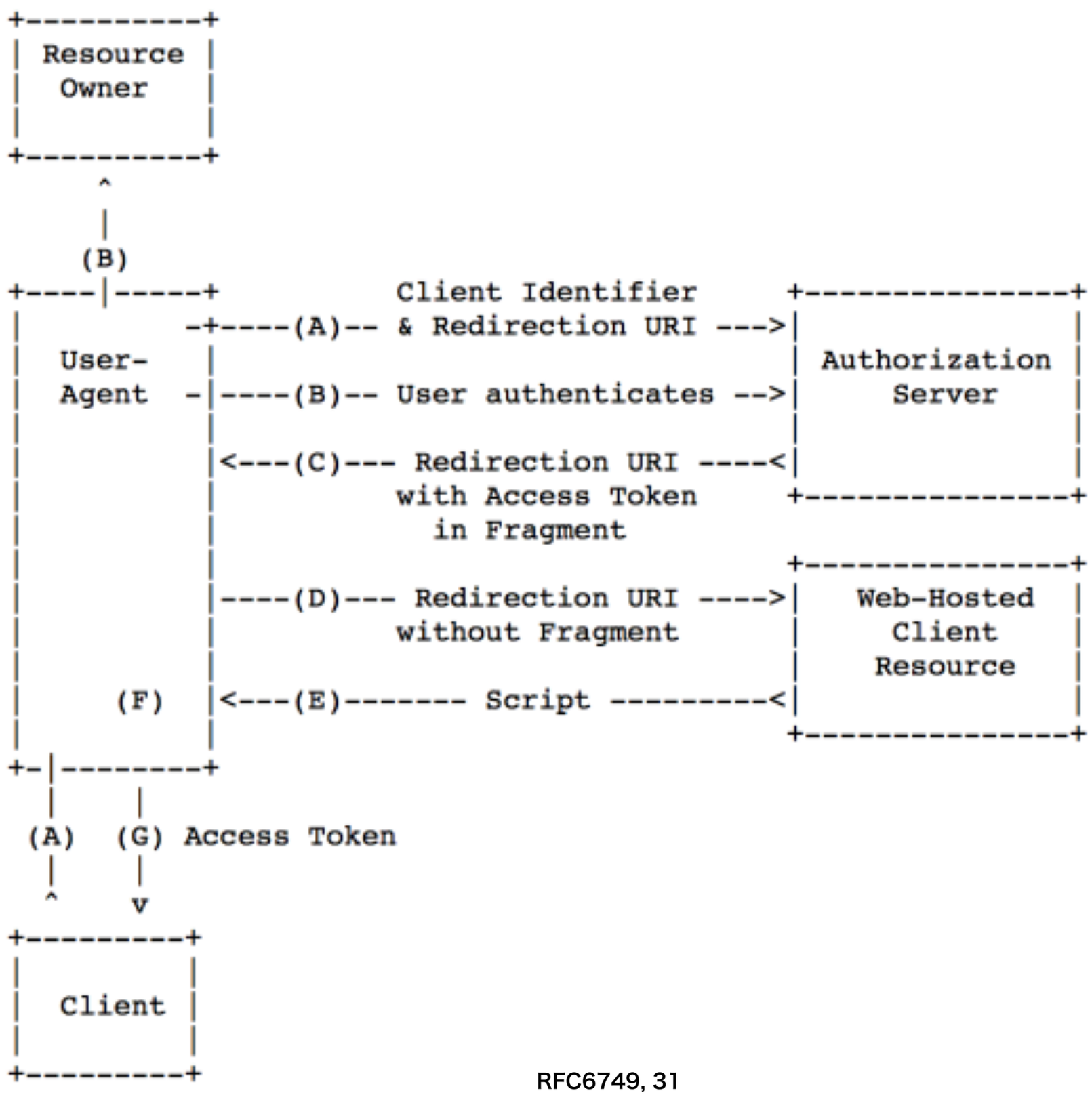
OAuth 2.0 for Native Application

- **RFC 8252** extends RFC 6759
- released 2017/10
- Android-AppAuth
  - https://github.com/openid/AppAuth-Android

OAuth 2.0 before RFC8252

- Client is **un-trusted to store secret**
  - APK Reverse Engineering
  - Reading by Web Proxy
- Easy to get spoofed
- **Implicit Flow** is the way to go

Implicit Flow



RFC6749, 31

Problem with Implicit Flow

- Unfriendly Token Refresh
- Stealing the Redirect
- Embedded User-Agent (WebView)

Problem with Implicit Flow

- **Unfriendly Token Refresh**
- Stealing the Redirect
- Embedded User-Agent (WebView)

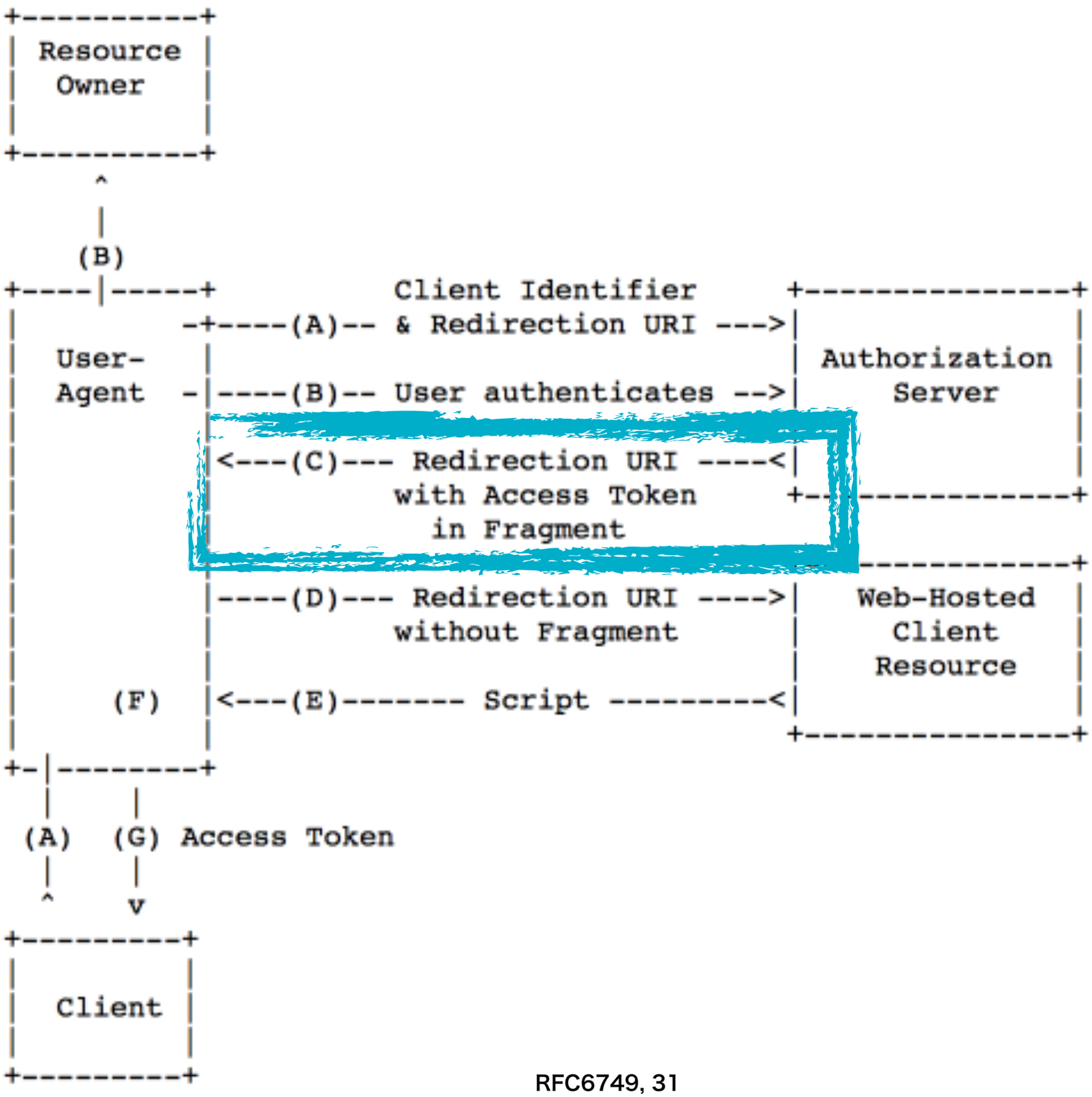
Problem with Implicit Flow

- **Unfriendly Token Refresh**
  1. Implicit Flow does not support Refresh Token
  2. Cannot refresh token without user interaction

Problem with Implicit Flow

- **Stealing the Redirect**
- Unfriendly Token Refresh
- Embedded User-Agent (WebView)

Stealing the Redirect



RFC6749, 31

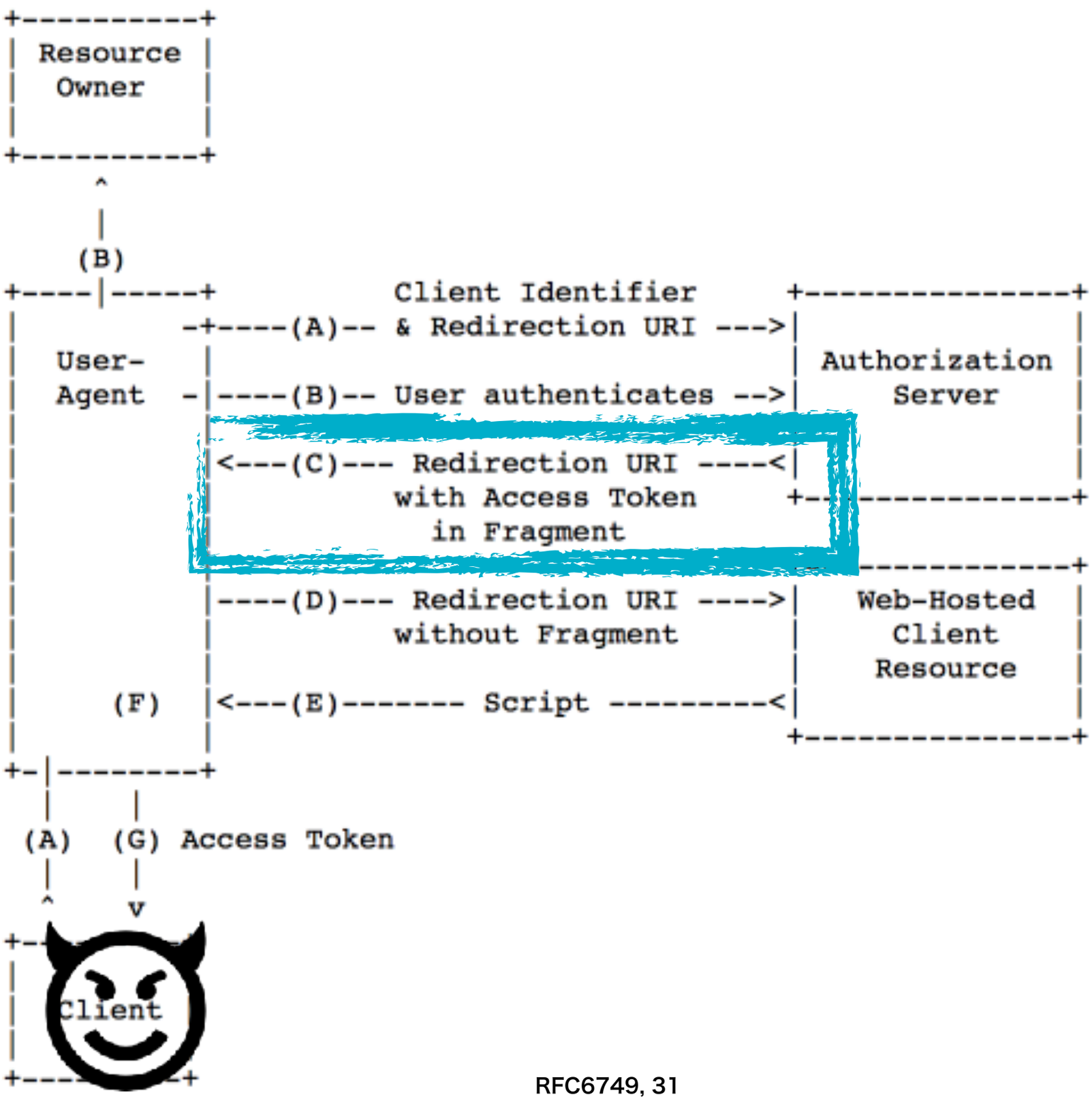
Defining Custom URL Scheme

```
<intent-filter>
  <action android:name="android.intent.action.VIEW"
  <category>
    android:name=“android.intent.category.DEFAULT”
  </category>
  <category>
    android:name=“android.intent.category.BROWSABLE
  </category>
  <data>
    android:scheme="com.ken5scal.ex"
    android:host=“oauth2redirect”
  </data>
</intent-filter>
```

Sending a Request w/ custom URL Scheme

```
Uri.parse("https://ex.com/authZ_ep").buildUpon()
  .appendQueryParameter("response_type", "code")
  .appendQueryParameter("client_id", ${client_id})
  .appendQueryParameter("scope", "profile")
  .appendQueryParameter(“state", “xxxxxxxxxxx")
  .appendQueryParameter(
    "redirect_uri", "com.ken5scal.ex:/oauth2redirect
```

Redirect can be acquired by EVIL



RFC6749, 31

Problem with Implicit Flow

- **Stealing the Redirect**
  1. Cannot closely tie Redirect URL with actual App.
  2. Token gets likely to be stolen by spoofing

Problem with Implicit Flow

- Stealing the Redirect
- Unfriendly Token Refresh
- **Embedded User-Agent (WebView)**

Presumption

- **Only WebView** option when OAuth 2.0 came out

Obtaining Plain-txt sensitive Info from WebView

```
var webView = object : WebView(baseContext) {
  override fun dispatchKeyEventPreIme(
   event: KeyEvent?): Boolean {
     Log.d(“even_password", event?.keyCode.toString()
     return super.dispatchKeyEventPreIme(event)}}
```

Stealing Cookie

```
webView.webViewClient = object : WebViewClient(){
  override fun onPageFinished(view: WebView?, url:
  String?) {
    var cookieManager =
    CookieManager.getInstance().getCookie(url)
    for (cookie in cookieManager.split(";") ) {
      //You can steal cookie here
    }
}
```

No Address Bar

- User cannot verify **SSL Cert**
- Bad for **User Enlightenment**

Unfriendly Sign-in Experience

- **Session cannot be shared** with other App
- Anti-**SSO**

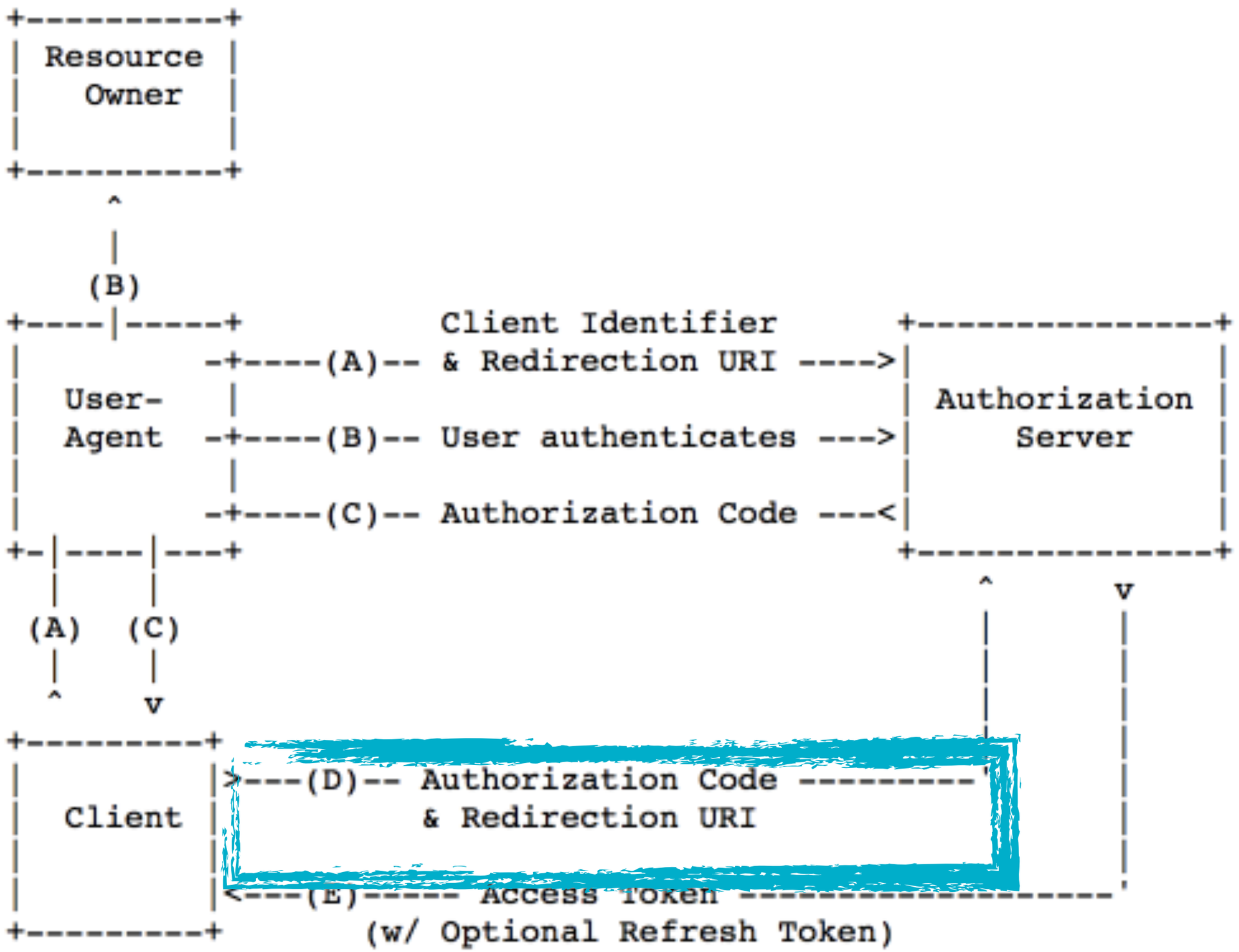
How to Address?

- **Authorization Code Grant Flow (Kinda)**
  - Stealing the Redirect
  - Unfriendly Token Refresh
- **External User-Agent**
  - Embedded User-Agent (WebView)

for Native Apps

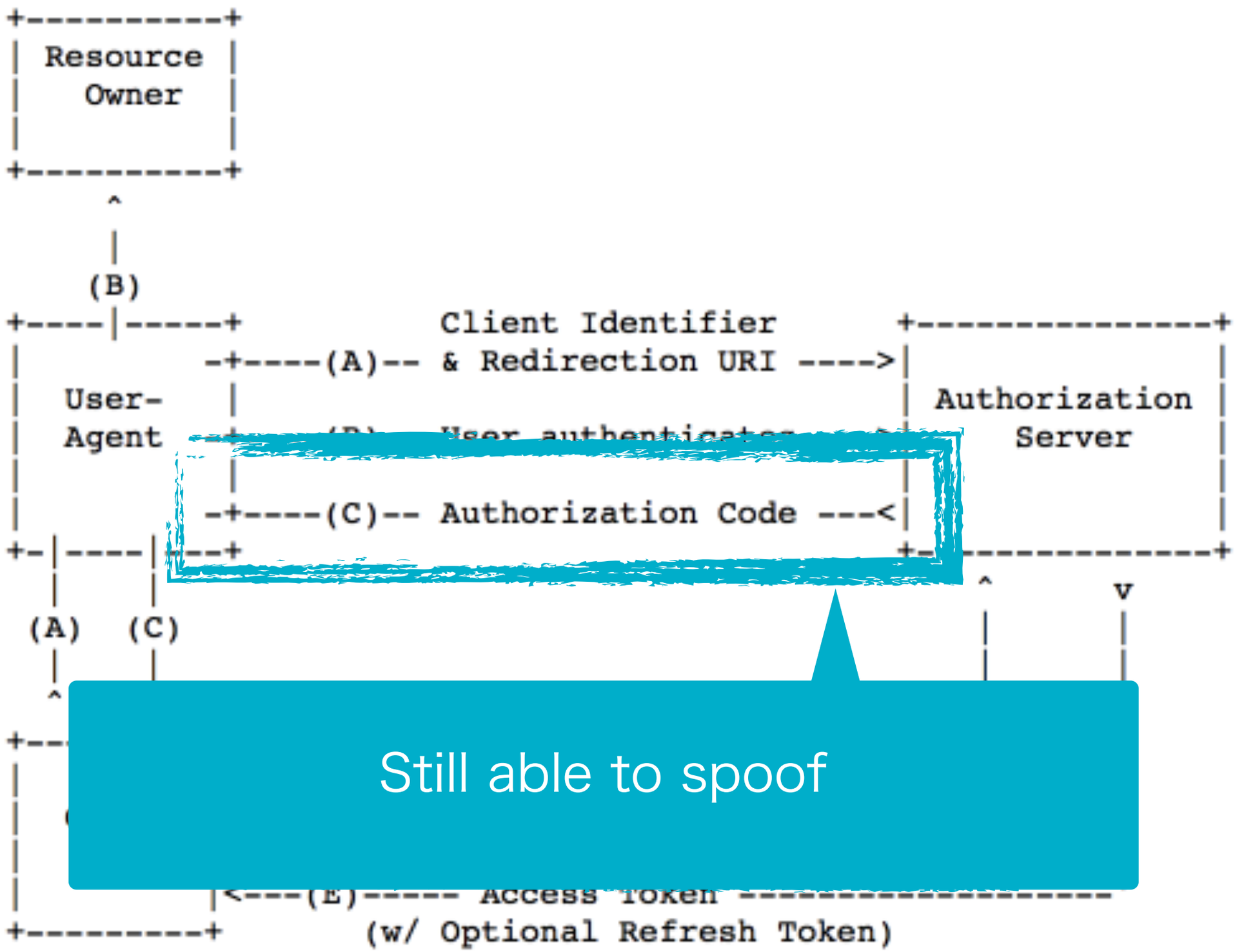
(RFC 8252)

Authorization Code Grant



RFC6749, 32

Authorization Code Grant

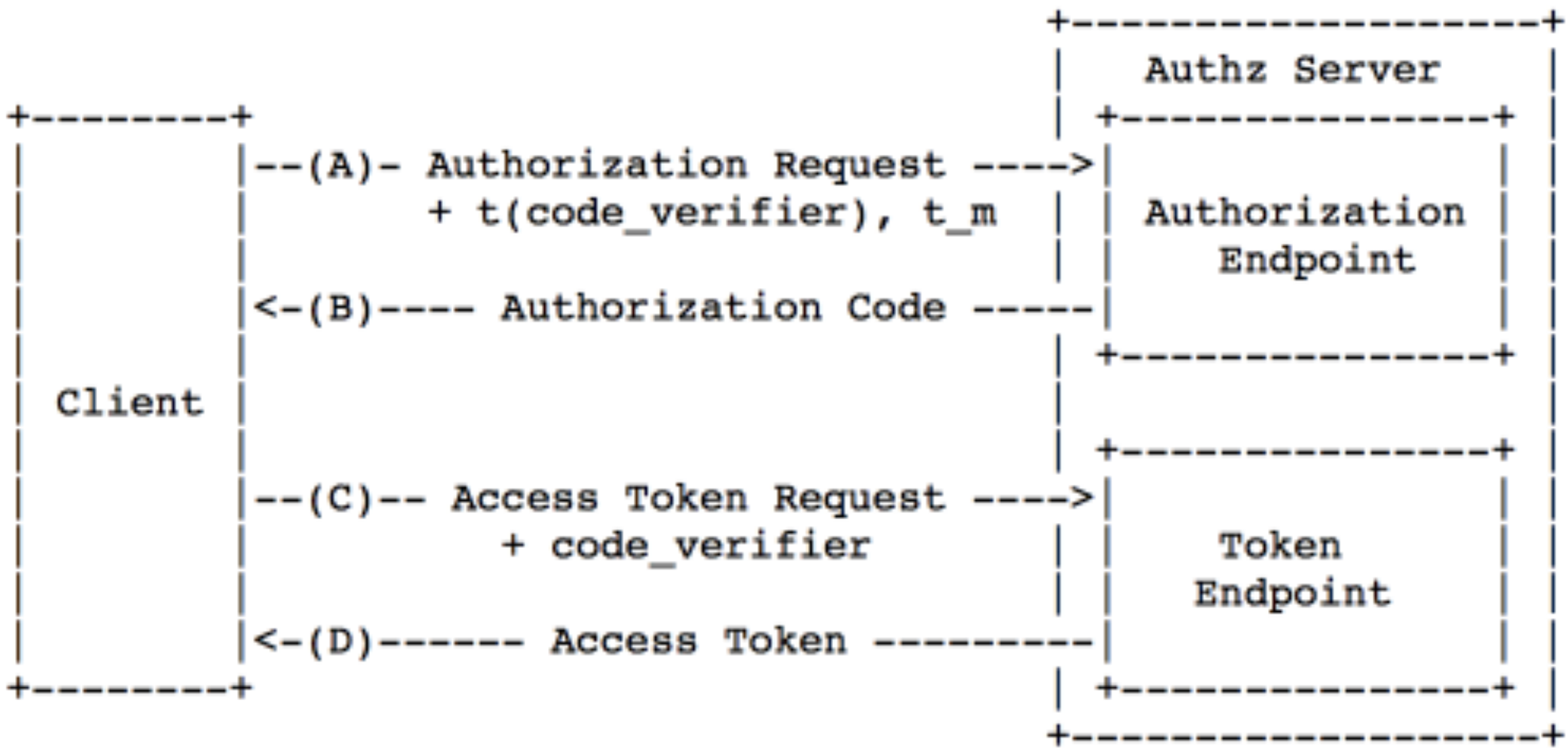


Still able to spoof

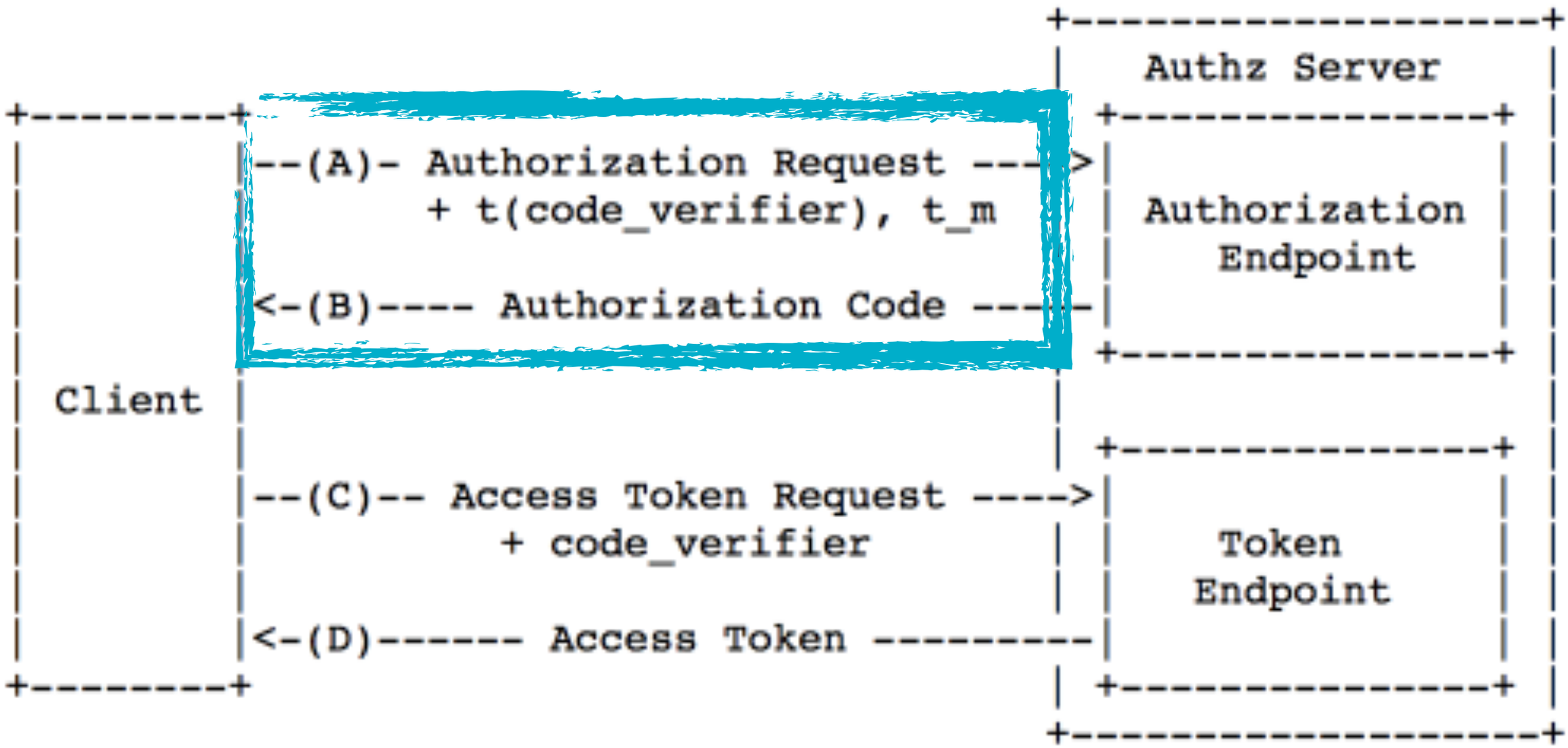
RFC6749, 32

PKCE (RFC 7636)

Proof Key for Code Exchange(PKCE) Overview



Generate Code Verifier



Generate Code Verifier

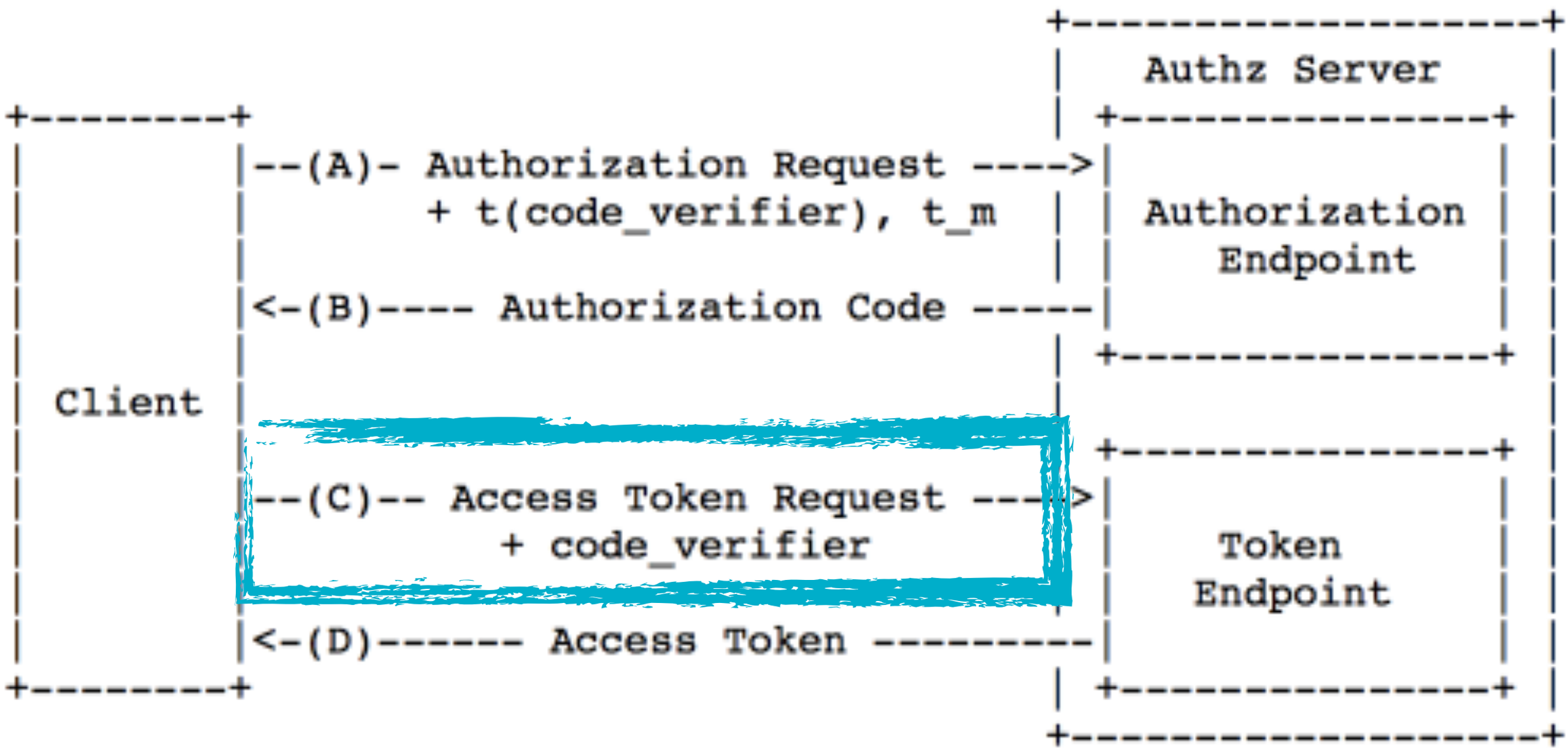
```
var srnm = SecureRandom()
var codeVerifier = byteArrayOf(64)
srnm.nextBytes(randomBytes)

var codeChallenge =
Base64.encodeToString(codeVerifier, Base64.NO_WRAP
or Base64.NO_PADDING or Base64.URL_SAFE)
```

Send Code Verifier with

```
Uri.parse(“https://ex.com/authZ_ep”).buildUpon()
  .appendQueryParameter("response_type", "code")
  .appendQueryParameter("client_id", $client_id)
  .appendQueryParameter(
    “redirect_uri", "com.ken5scal.ex:/oauth2redirect
.appendQueryParameter("scope", "profile")
.appendQueryParameter("state", "xxxxxxxxxxxxxxx")
.appendQueryParameter(
  "code_challenge", codeChallenge)
.appendQueryParameter(
  "code_challenge_method", "S256")
```

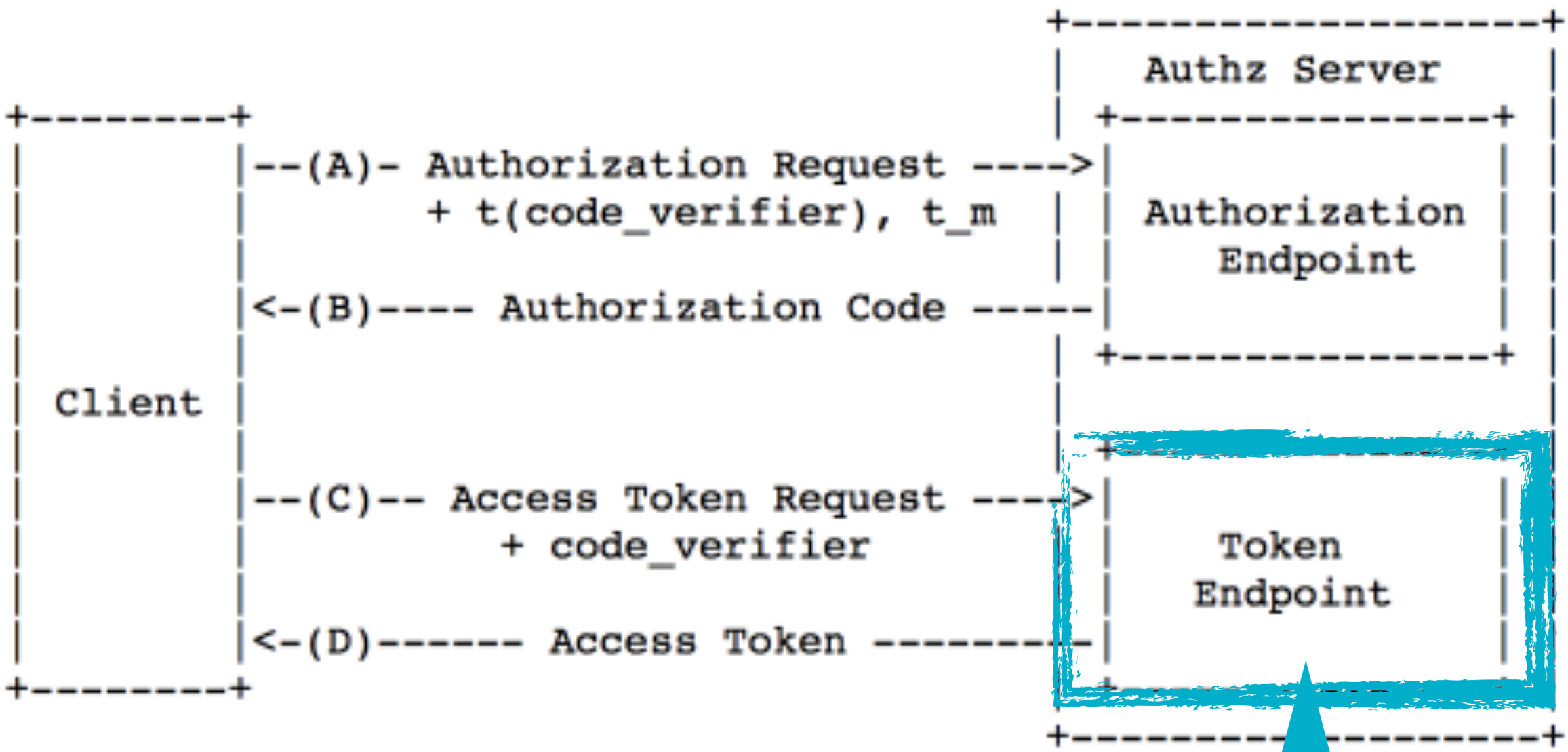
Access Token Request with Code Verifier



Access Token Request with Code Verifier

```
Uri.parse(“https://ex.com/token_ep").buildUpon()
  .appendQueryParameter("response_type", “code")
  .appendQueryParameter(“code”,$auth_code)
.appendQueryParameter(“client_id”,$client_id)
.appendQueryParameter(
   "redirect_uri", “com.ken5scal.ex:/oauth2redirect")
.appendQueryParameter("code_verifier",$codeVerifier)
```

Verify the challenge on Auth Server



1) do t(code_verifier) at Endpoint
2) compare the result with AuthReq

How to Address?

- Stealing the Redirect
  - > **PKCE**
- Unfriendly Token Refresh
  - > **AuthZ Code Grant Flow = refresh token is available**

Important Note

the use of the Implicit Flow with native apps is **NOT RECOMMENDED.**

Using App Link

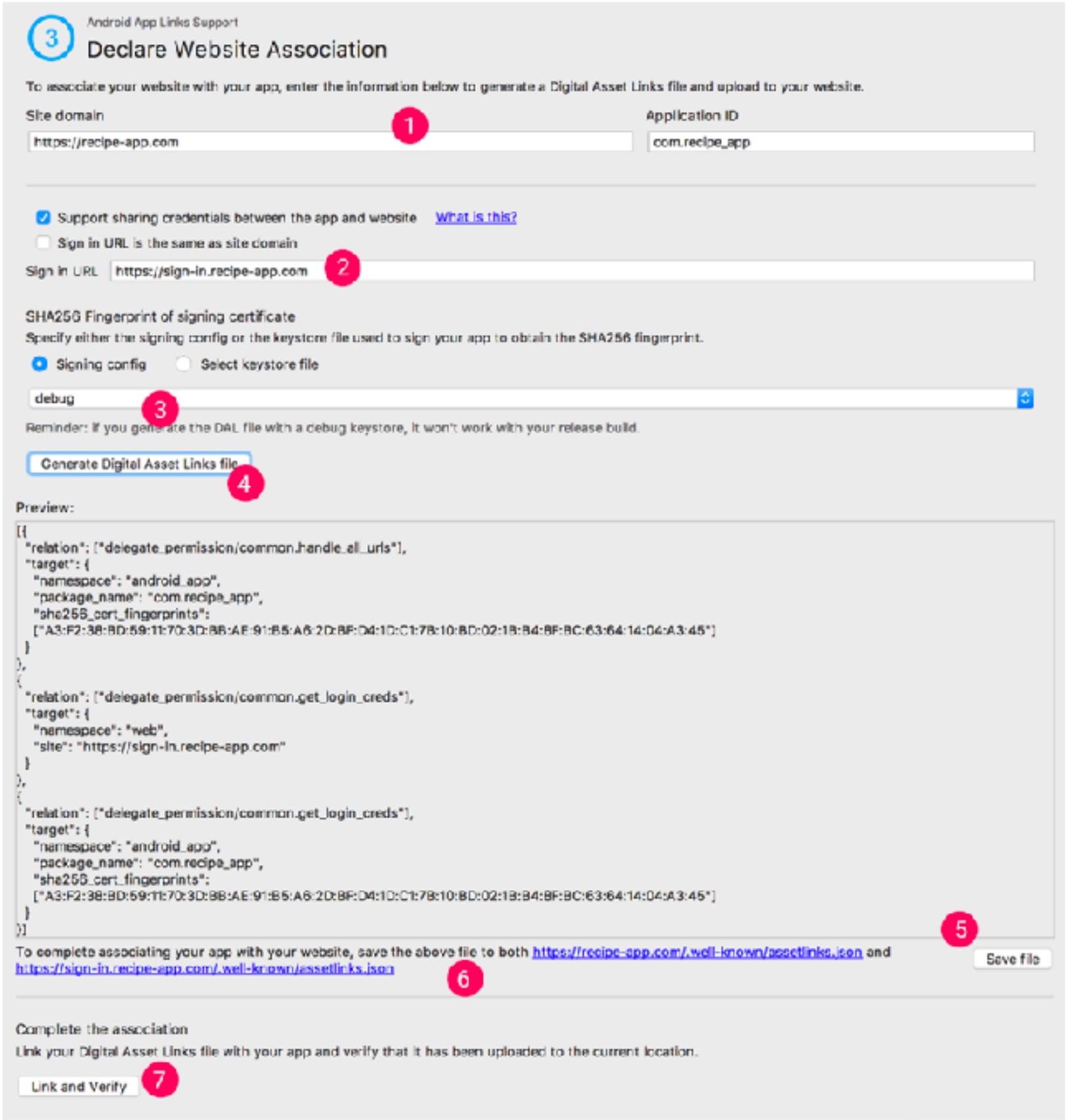
Alternative Approach for Redirect URL stealing

- "https" scheme can be set as Redirect URL
- For Android M or later, using **App Link is Recommended**
  - as it can tightly couple actual app and URL.
- *"App-claimed "https" scheme redirect URIs have some advantages compared to other native app* **redirect options in that the identity of destination app is guaranteed** *to the authorization server by the operating system. For this reason,* ***native apps SHOULD use them over the other options where possible***.

Intent-filter for App Link

```
<activity
android:name=“com.ken5scal.ex.AuthResponseReceiverAc
tivity">
  <intent-filter android:autoVerify="true">
    <action android:name="android.intent.action.VIEW"
    <category
    android:name="android.intent.category.DEFAULT"/>
    <category
    android:name="android.intent.category.BROWSABLE"/
    <data android:scheme="http"
        android:host="ex.ken5scal.com"
        android:pathPattern="/oauth2redirect"/>
    <data android:scheme="https" />
  </intent-filter>
</activity>
```

Linking App and WebSite



- **RFC insists to do PKCE** anyway because risk remains
- *"but the app is still a public client; further, the URI is sent using the operating system's URI dispatch handler with unknown security properties.* (OAuth 2.0 for Native Apps, p10)"

External User-Agent

native apps **MUST NOT** use embedded user-agents to perform authorization requests

RFC8252, 14

In case of Google…

Modernizing OAuth interactions in Native Apps for Better Usability and Security

Monday, August 22, 2016

The rollout schedule for the deprecation of web-views for OAuth requests to Google is as follows. Starting **October 20, 2016**, we will prevent new OAuth clients from using web-views on platforms with a viable alternative, and will phase in user-facing notices for existing OAuth clients. On **April 20, 2017,** we will start blocking OAuth requests using web-views for all OAuth clients on platforms where viable alternatives exist.

https://developers.googleblog.com/2016/08/modernizing-oauth-interactions-in-native-apps.html

Use External User-Agent

- Chrome Custom Tab
- Google Sign In SDK
  - Becareful w/ External User-Agents
  - Use only trusted SDK

Other Security Consideration

Other Security Consideration

- Registration of Native App Clients
- Cross-App Request Forgery Protections

Registration of Native App Clients

← クライアント ID の作成

アプリケーションの種類
- ○ ウェブ アプリケーション
- ● Android 詳細
- ○ Chrome アプリ 詳細
- ○ iOS 詳細
- ○ PlayStation 4
- ○ その他

名前

Android クライアント 4

署名証明書フィンガープリント
パッケージ名と SHA-1 署名証明書フィンガープリントを追加し、使用を Android アプリに制限します。
AndroidManifest.xml ファイルからパッケージ名を取得し、次のコマンドを実行してフィンガープリント

```
$ keytool -exportcert -keystore path-to-debug-or-production-keystore -list -v
```

12:34:56:78:90:AB:CD:EF:12:34:56:78:90:AB:CD:EF:AA:BB:CC:DD

パッケージ名
AndroidManifest.xml ファイル内

com.example

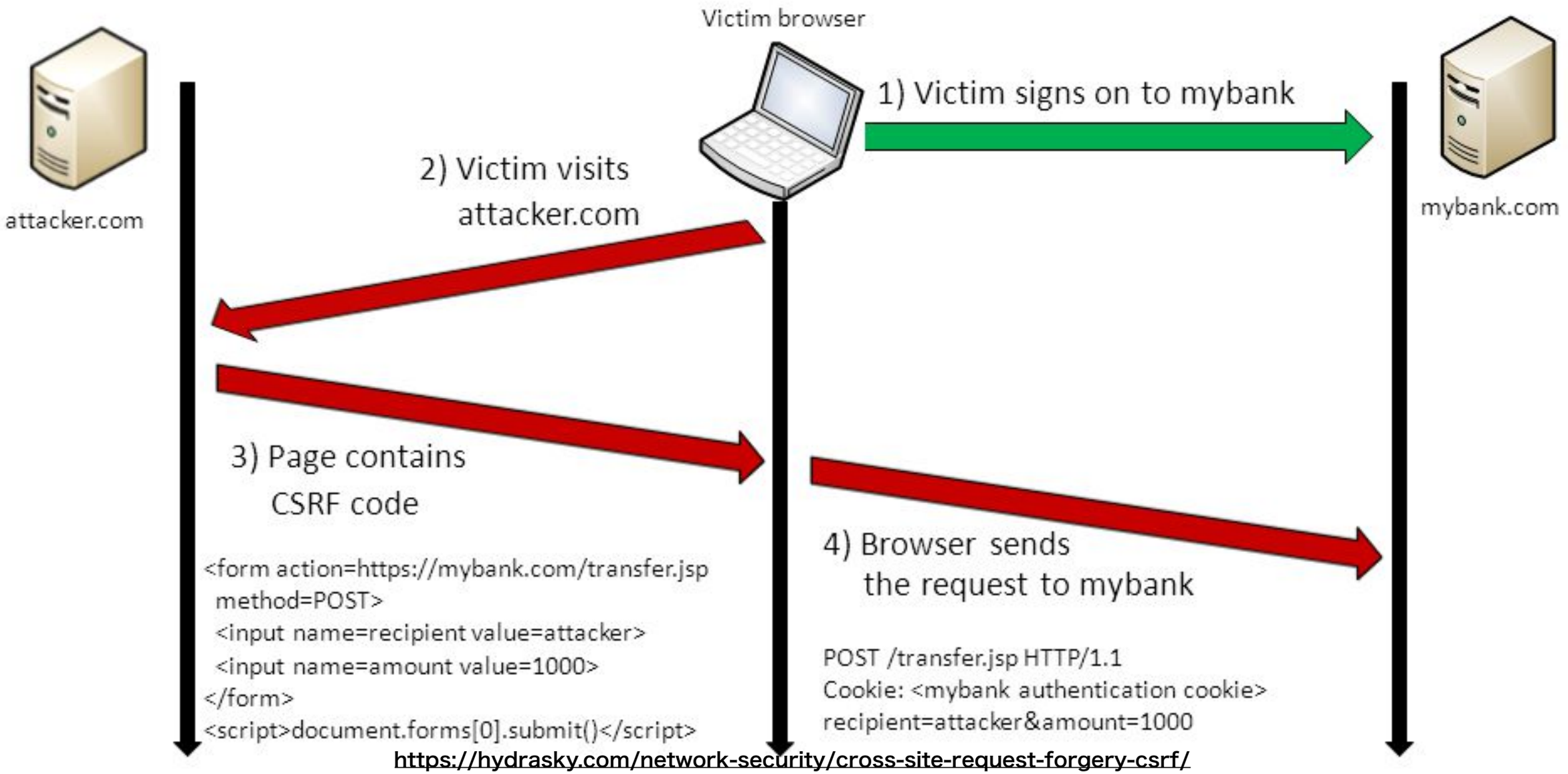
作成 キャンセル

Cross-App Request Forgery Protection

- Protecting CSRF in Native App.
- CSRF is type of attack “that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.”
- https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

CSRF in nutshell

Cross-Site Request Forgery (CSRF)



https://hydrasky.com/network-security/cross-site-request-forgery-csrf/

Cross-App Request Forgery Example

- Attacker creates S3 bucket
- Attacker makes a AuthZ request and receives response
- Redirect the response to Target App.
- User receives token (or auth_code)
- User will uploads her/his data on Attacker's S3 bucket
- **Granting access to unsafe resource**

State Parameter for CARF

- Make something like CSRF Token: **State Parameter**
- App creates a random value and send to server w/ AuthZ request

```
var srnm = SecureRandom()
var randomBytes = byteArrayOf(64)
srnm.nextBytes(randomBytes)
var state = Base64.encodeToString(randomBytes,
Base64.NO_WRAP or Base64.NO_PADDING or
Base64.URL_SAFE)
```

- Auth server sends back the value with AuthZ response
- App checks whether state parameters are the same.
- If they are the same , they are in same session and safe
- **Recommended in RFC 6749 and RFC 8252**

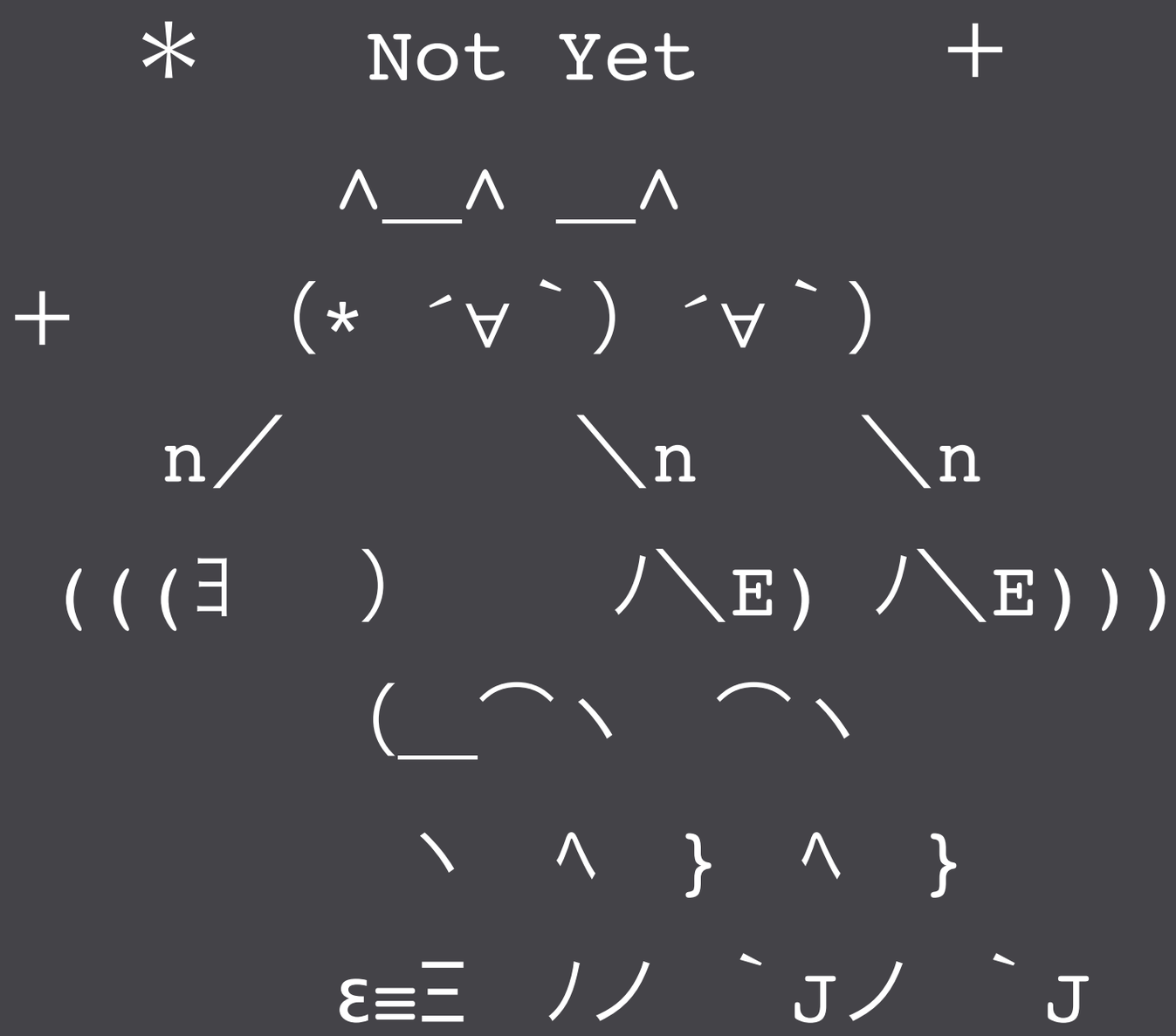
for Native Apps

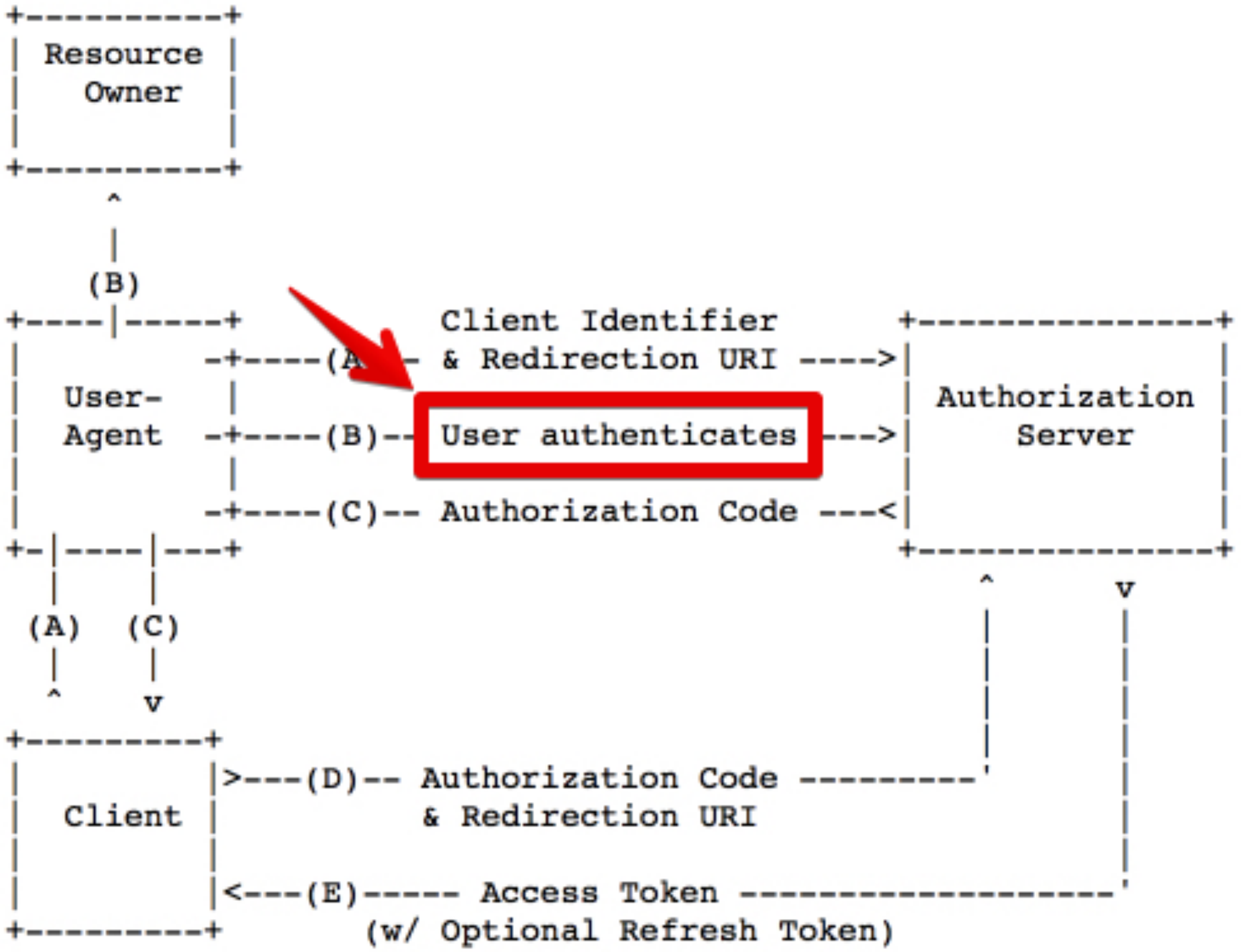
Summary

OAuth2.0 for Android Summary

- Follow RFC 8252
- AppAuth-Android is Good
  - https://github.com/openid/AppAuth-Android
- Use
  - PKCE
  - Authorization Grant Type
  - External User-Agent

Is API Protected?


* Not Yet +
^__^ __^
+ (* ´∀`) ´∀`)
n/ \n \n
(((∃) 人E) 人E)))
(_⌒ヽ ⌒ヽ
ヽ ^ } ^ }
ε≡Ξ ノノ `Jノ `J


Resource Owner
(B)
User-Agent
(A)
Client Identifier & Redirection URI
User authenticates
Authorization Server
(C)
Authorization Code
Client
(D)
Authorization Code
& Redirection URI
(E)
Access Token
(w/ Optional Refresh Token)

- Client Identifier
- & Redirection URI -

User authenticates

- Authorization Code

Auth君の名はdroid

FIDO™ alliance

for Android

FIDO in Nutshell

What is FIDO

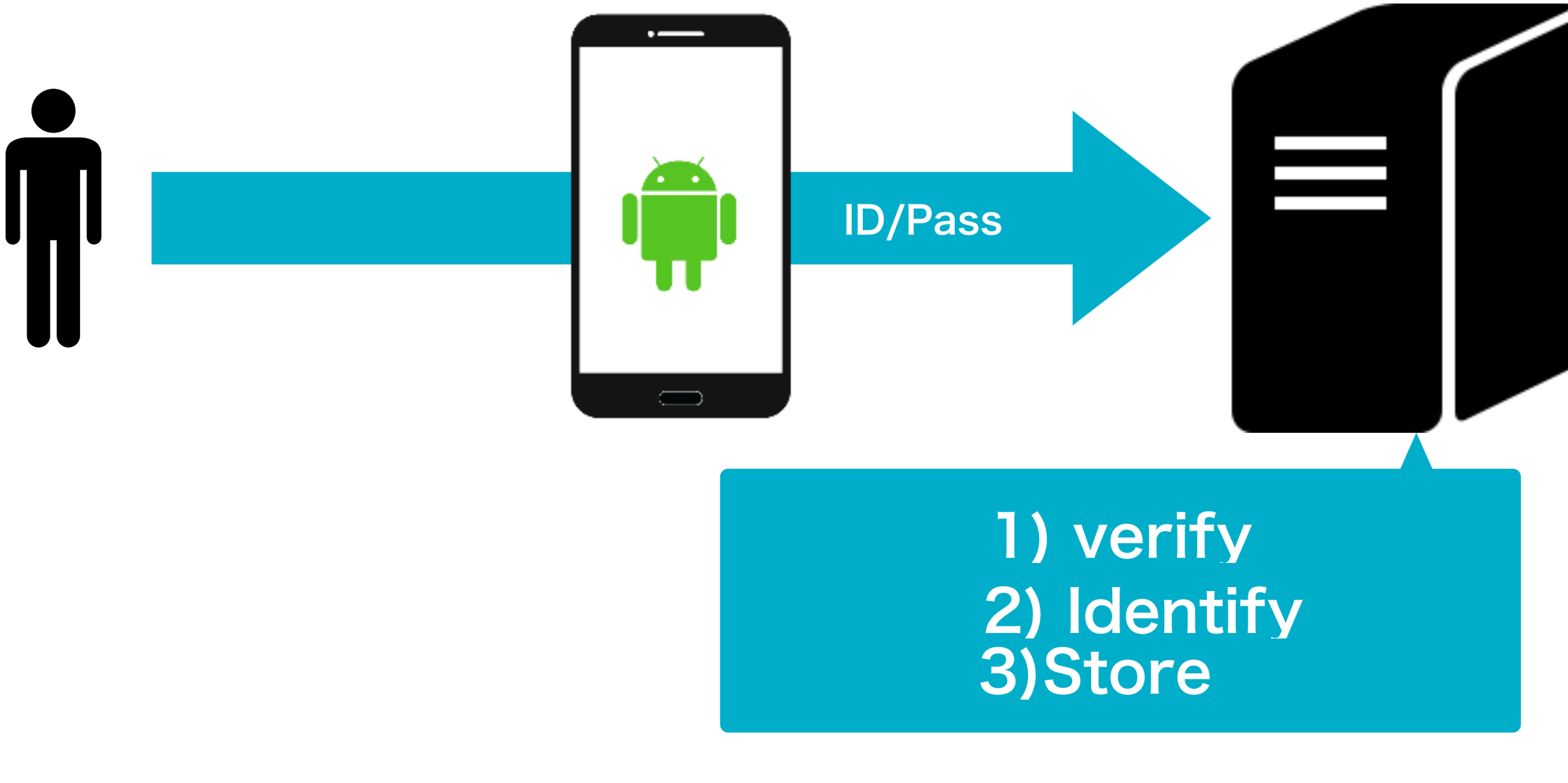
- Fast IDentity Online Alliance
  - Less relying on Password for AuthN
  - Scalable, Inter-operable, Open standard AuthN
- Boards
  - Google, Facebook, Amazon, Microsoft
  - LINE, MUFJ(Bank), Docomo

UAF vs U2F

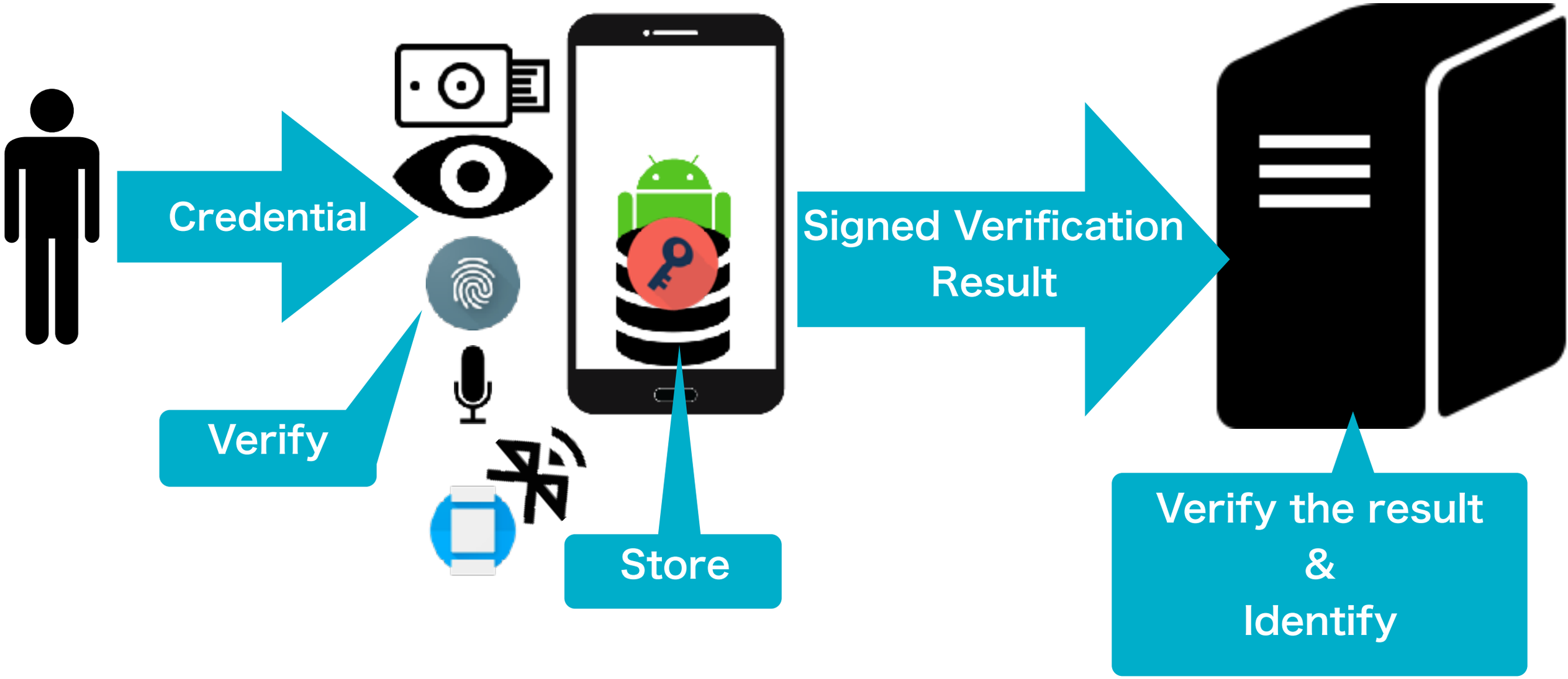
Legacy AuthN: The Password

- Not to mention it's bad **UX**
- Not good in term of the system
  - Easily targeted by phishing attack.
    - Chance of getting breached
- Too much responsibility on Server-side

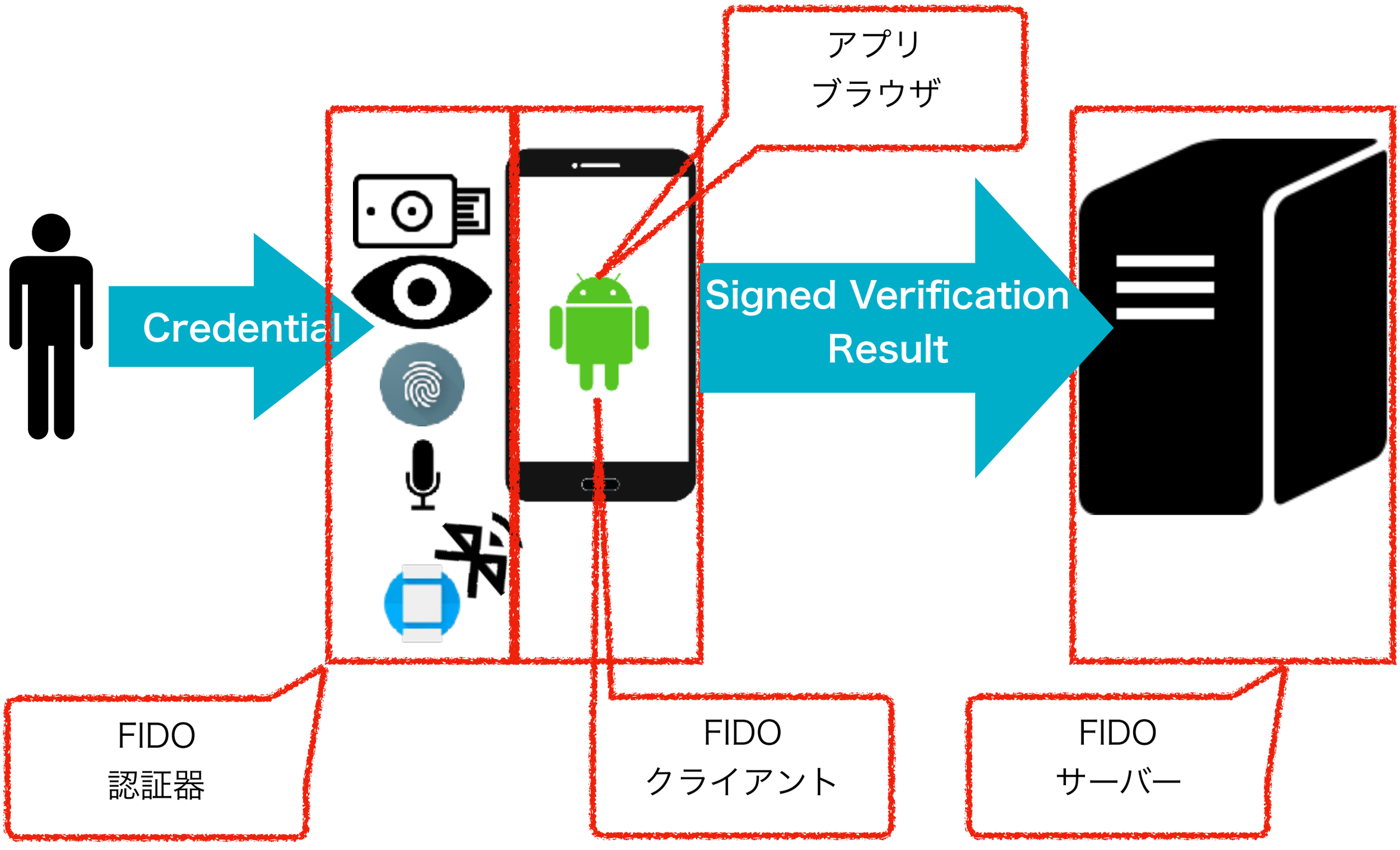
Server-Side Responsibility



FIDO AuthN Model

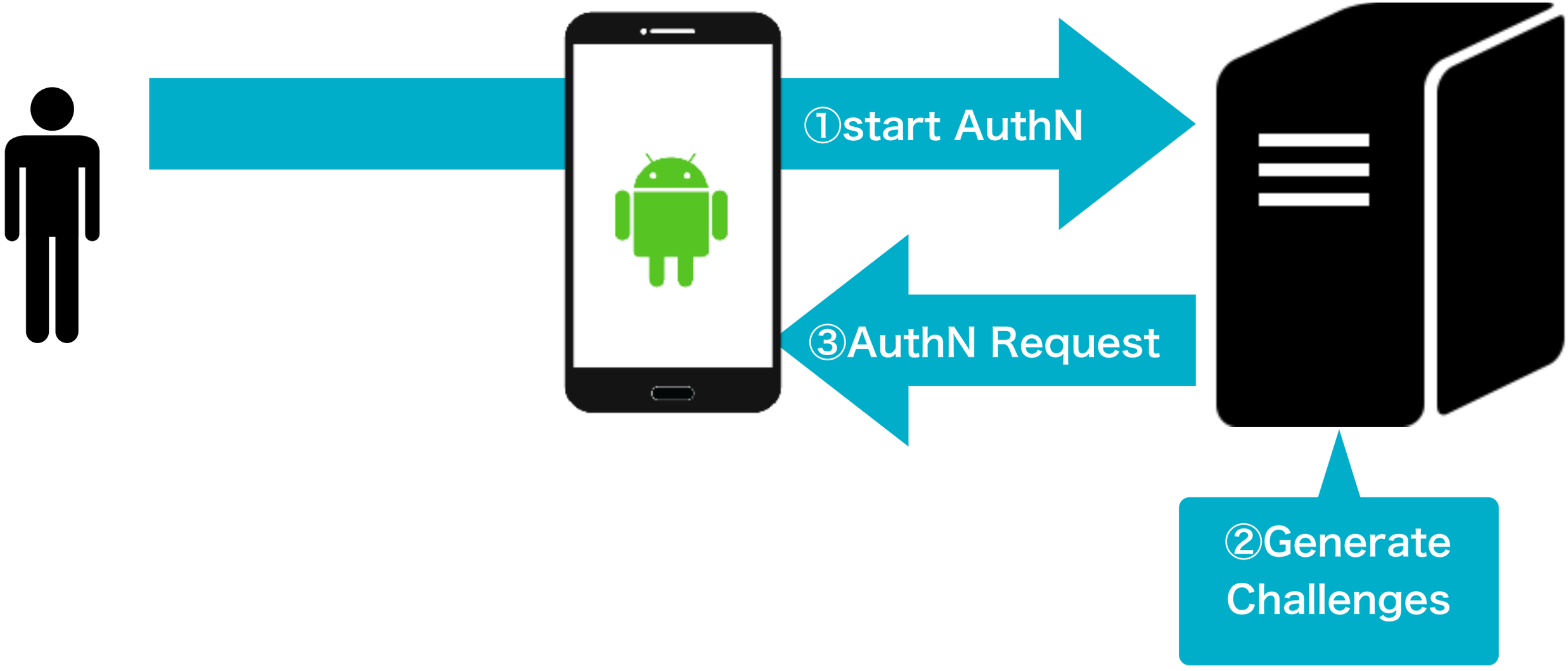


Roles in FIDO



AuthN Flow

AutheN Flow



Authentication Request

```
[{
"header": {
    "upv": { "major": 1, ”minor": 1 },
    "op": "Auth",
    "appID": "https://hoge.com/SampleApp/uaf/facets",
    "serverData": "xxxxxxxxxxxxxxxxx"
},
"challenge": “this_is_challenge”,
"policy": {
    "accepted": [ … ],
    "disallowd": [ … ]
}
```

Authentication Request

```
[{
"header": {
    "upv": { "major": 1, ”minor": 1 },
    "op": "Auth",
    "appID": "https://hoge.com/SampleApp/uaf/facets",
    "serverData": "xxxxxxxxxxxxxxxx"
},
"challenge": “this_is_challenge”,
"policy": {
    "accepted": [ … ],
    "disallowd": [ … ]
}
```

Authentication Request

```
[{
"header": {
    "upv": { "major": 1, ”minor": 1 },
    "op": "Auth",
    "appID": "https://hoge.com/SampleApp/uaf/facets",
    "serverData": "xxxxxxxxxxxxxxxxx"
},
"challenge": “this_is_challenge”,
"policy": {
    "accepted": [ … ],
    "disallowd": [ … ]
}
```

Authentication Request

```
[{
"header": {
    "upv": { "major": 1, ”minor": 1 },
    "op": "Auth",
    "appID": "https://hoge.com/SampleApp/uaf/facets",
    "serverData": "xxxxxxxxxxxxxxxxx"
},
"challenge": “this_is_challenge”,
"policy": {
    "accepted": [ … ],
    "disallowd": [ … ]
}
```

Verification & Signing

⑤Verify & generate result

⑥Sign verification result by AuthN priv key

Signing

```
val ks = KeyStore.getInstance("AndroidKeyStore")
ks.load(null)
val privateKey = ks.getEntry("authN_key",null)
if (privateKey is KeyStore.PrivateKeyEntry) {
    val s = Signature.getInstance("SHA256withECDSA")
    s.initSign(privateKey.privateKey)
    s.update(authNResult)
    val assertionInByteArray = s.sign()
}
```

Sending Authentication Response


```
[{
"assertions": [{
  "assertion": "assertionInByteArray",
  "assertionScheme": "UAFV1TLV" }],
"fcParams": "I_will_explain_this_next_time",
"header": {
  "appID": "https://hoge.com/SampleApp/uaf/facets",
  "op": "Auth",
  "serverData": "xxxxxxxxxxxxxxxxx",
  "upv": { "major": 1, "minor": 1} }
}]
```

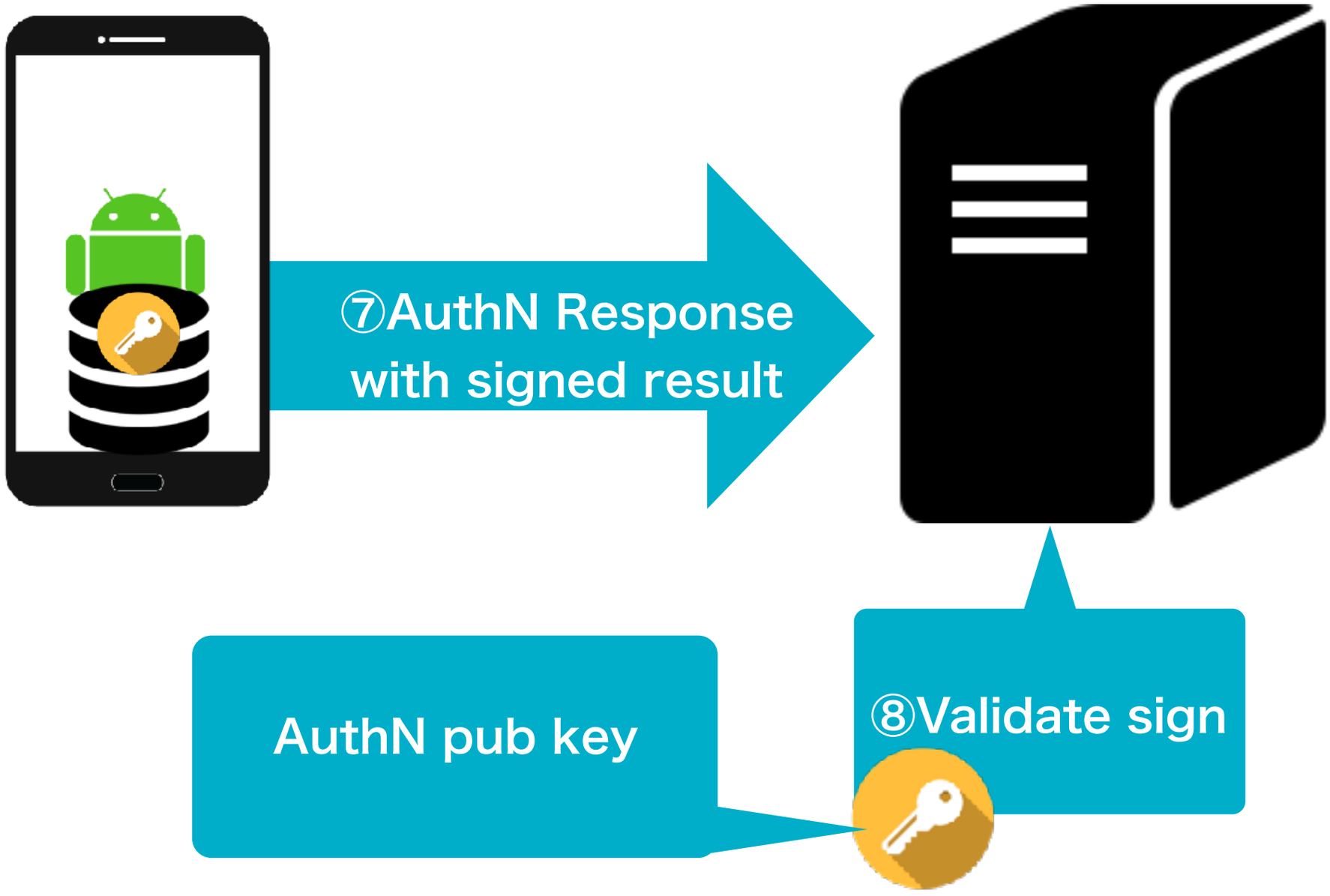
Sending Authentication Response


```
[{
"assertions": [{
   "assertion": "assertionInByteArray",
   "assertionScheme": "UAFV1TLV" }],
"fcParams": "I_will_explain_this_next_time",
"header": {
   "appID": "https://hoge.com/SampleApp/uaf/facets",
   "op": "Auth",
   "serverData": "xxxxxxxxxxxxxxxxx",
   "upv": { "major": 1, "minor": 1} }
}]
```

Sending Authentication Response

```
[{
"assertions": [{
  "assertion": "assertionInByteArray",
  "assertionScheme": "UAFV1TLV" }],
"fcParams": “challenge_response”,
"header": {
  "appID": "https://hoge.com/SampleApp/uaf/facets",
  "op": "Auth",
  "serverData": "xxxxxxxxxxxxxxxxx",
  "upv": { "major": 1, "minor": 1} }
}]
```

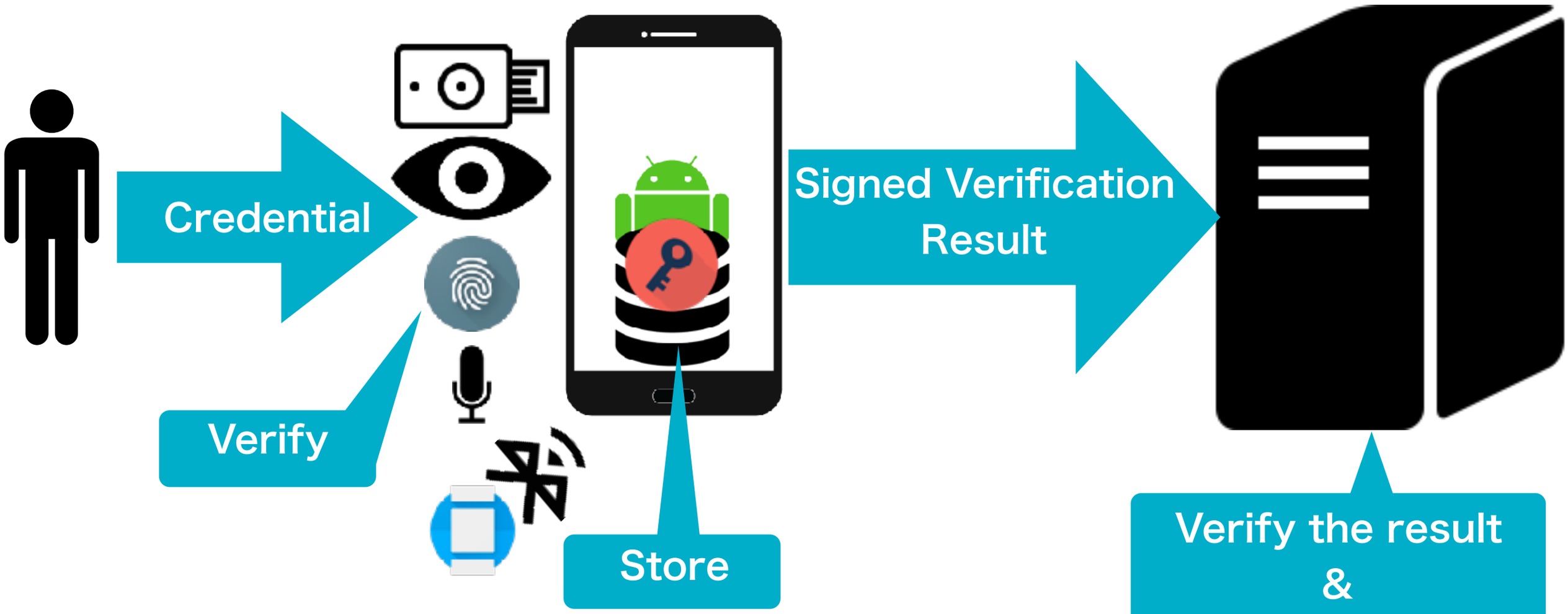
Sending Authentication Response



What does this achieve?

- **Authenticity** Verification
- **Integrity** Verification
- User **AuthN on Android** && Device **AuthN on Server**
  - **2FA by Default**

Revisiting FIDO AuthN



- Not to mention it's bad **UX**
- Easily targeted by phishing attack.
- Chance of getting breached
- Too much responsibility on Server-side

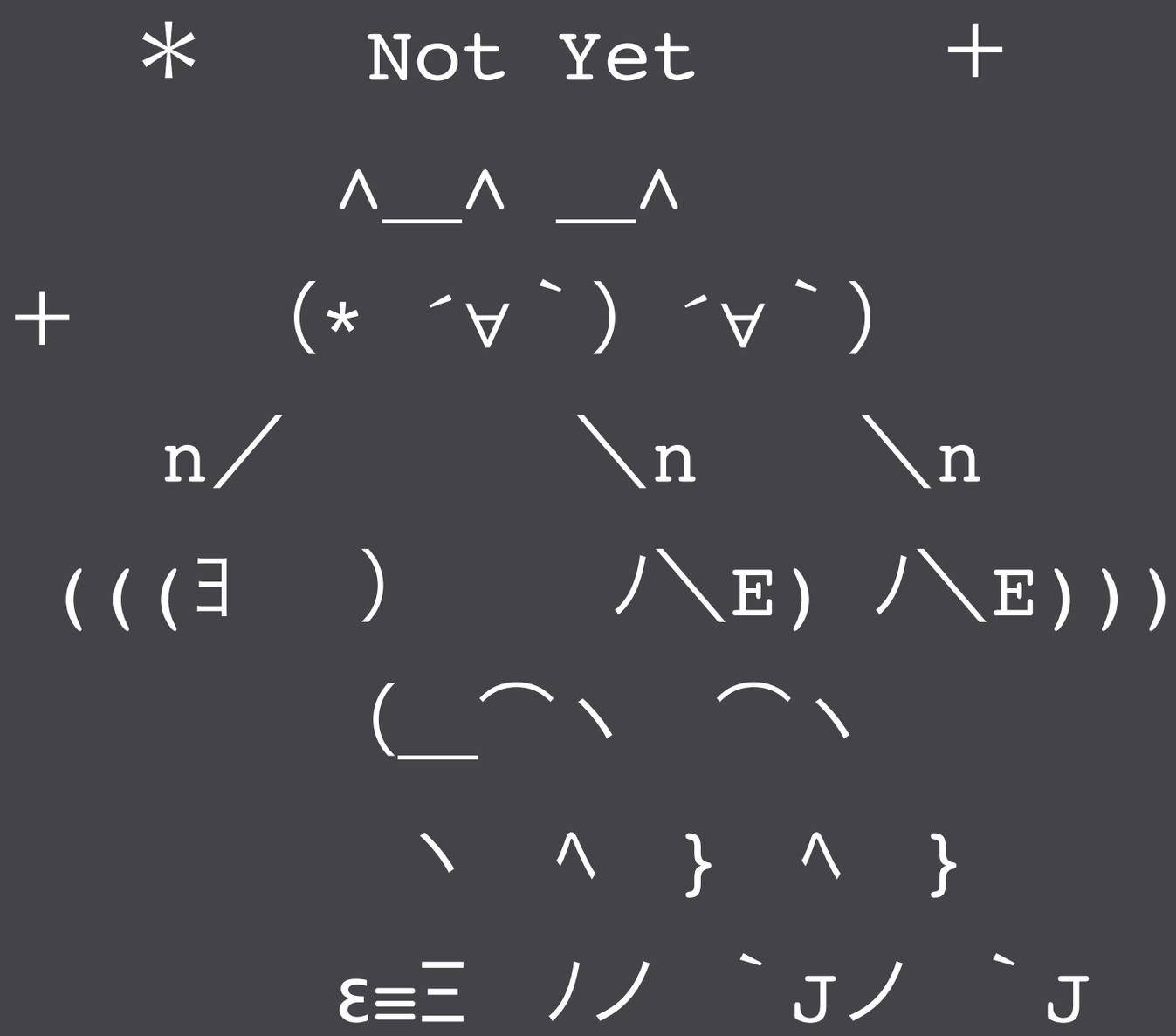
Pro-Tip: FIDO in Transaction Confirmation

Amazon Joins FIDO Board of Directors

Posted on January 23, 2018 by Alex Perala

https://mobileidworld.com/amazon-joins-fido-board-901231/

Is API Protected?


* Not Yet +
^__^ __^
+ (* ´∀`) ´∀`)
n/ \n \n
(((∃) 人E) 人E)))
(_⌒ヽ ⌒ヽ
ヽ ^ } ^ }
ε≡Ξ ノノ `Jノ `J

Private Key Security

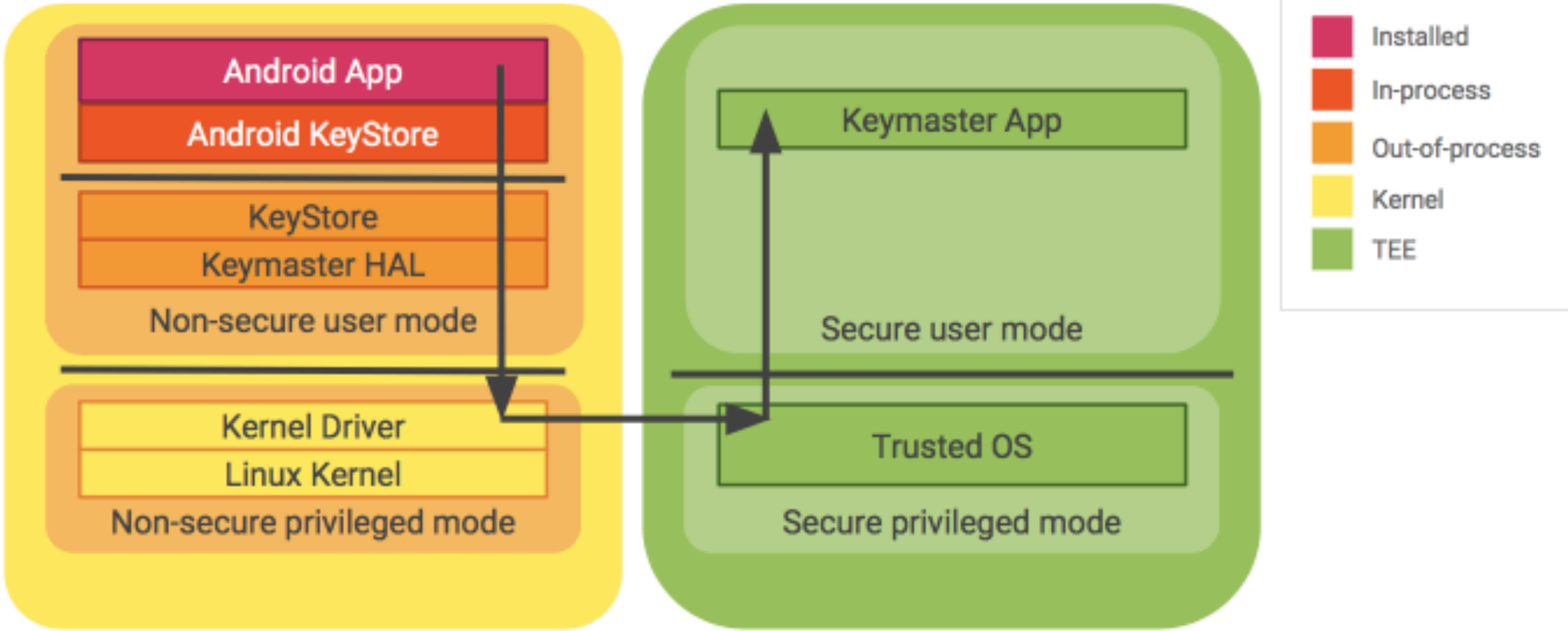
Secure Private Key

Protecting Private Key

- Is it **stored in safely manner**
- How to **verify storing method**

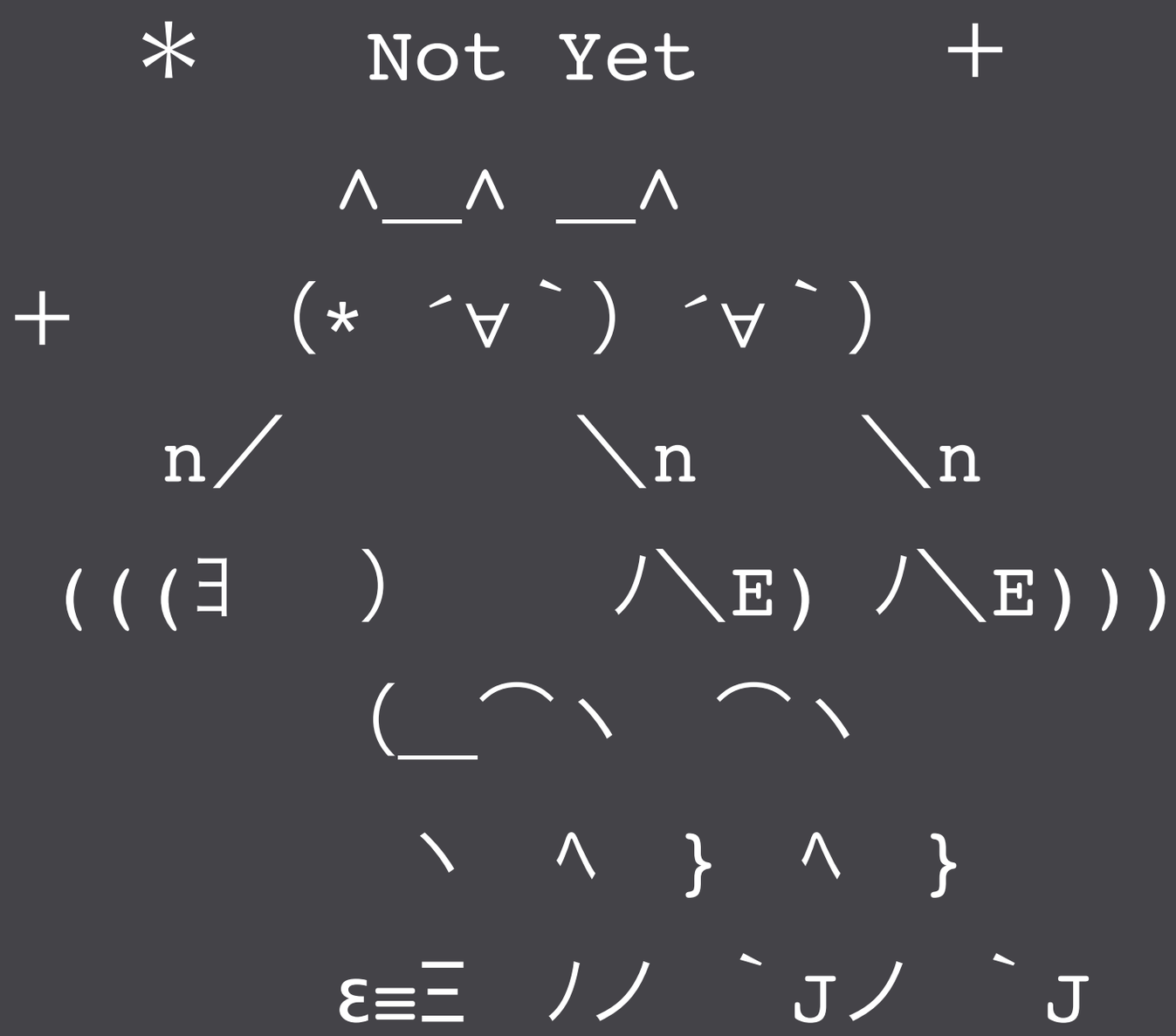
HW-backed KeyStore

KeyStore Architecture



Android Keystore Attestation

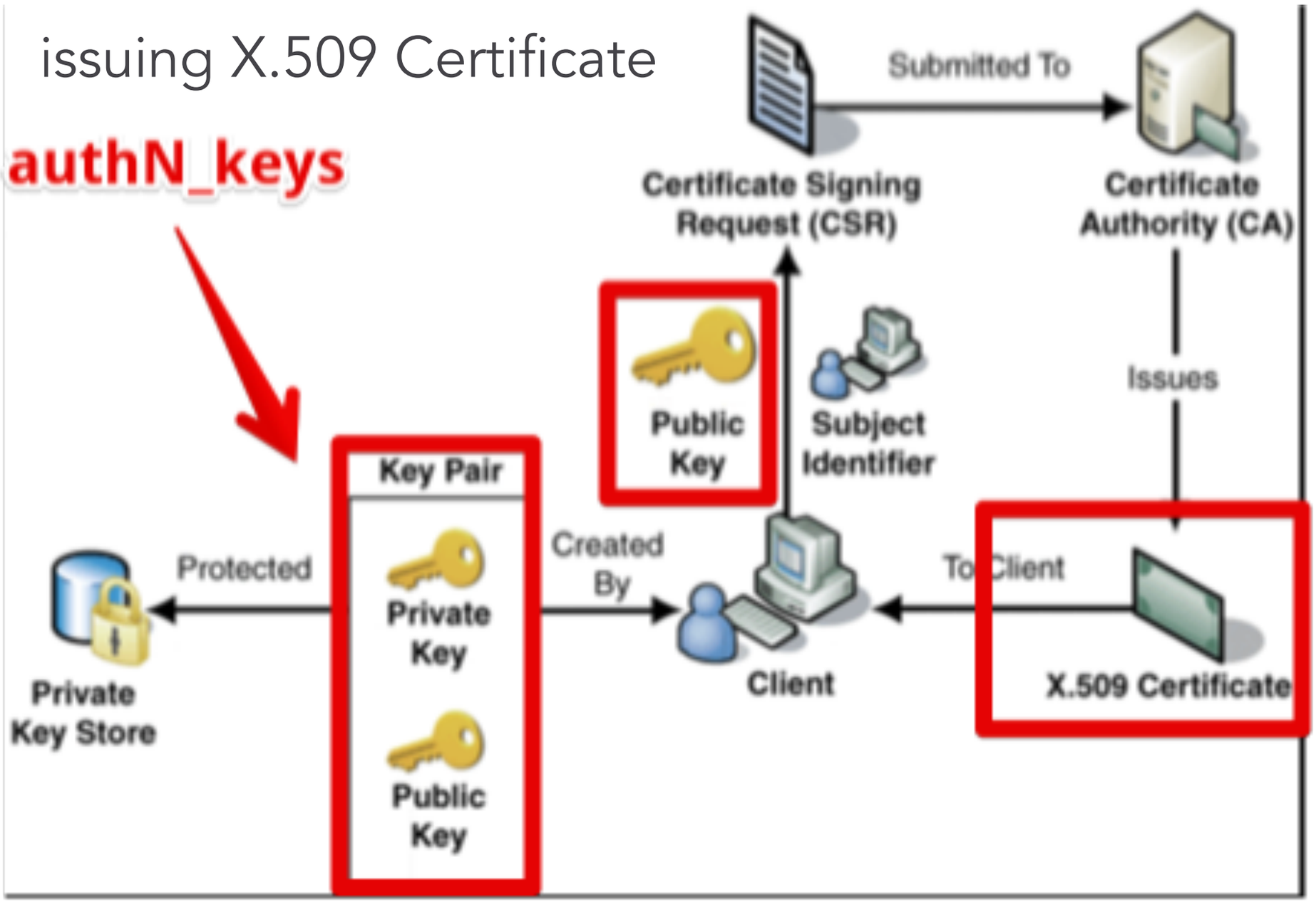
Is API Protected?


* Not Yet +
^__^ __^
+ (* ´∀`) ´∀`)
n/ \n \n
(((∃) 人E) 人E)))
(_⌒ヽ ⌒ヽ
ヽ ^ } ^ }
ε≡Ξ ノノ `Jノ `J

Key Attestation

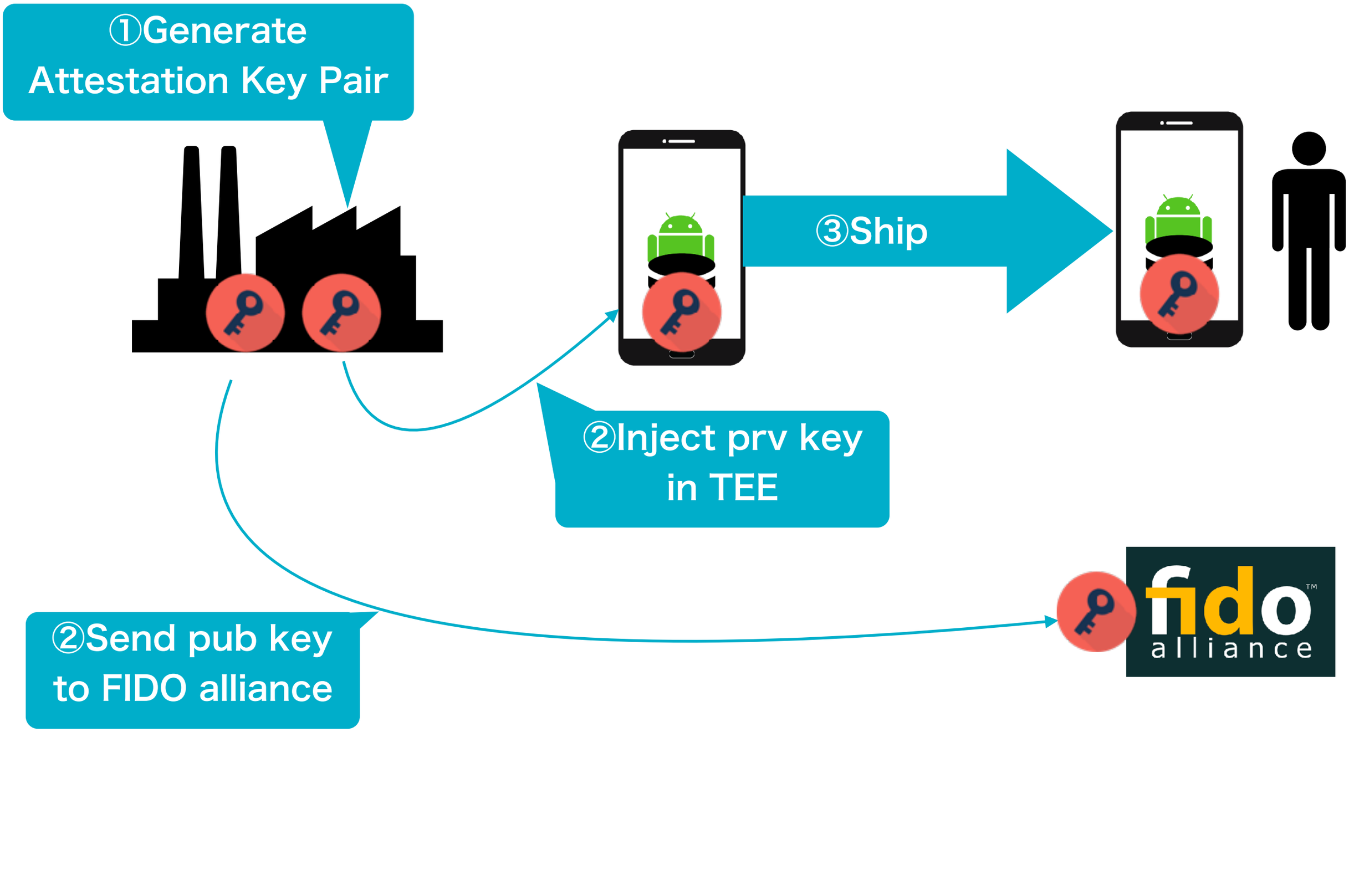
Key Attestation

- Ensure and certify keys by using CA private key in TEE
  - issuing X.509 Certificate



https://msdn.microsoft.com/en-us/library/ff647097.aspx

Key Attestation

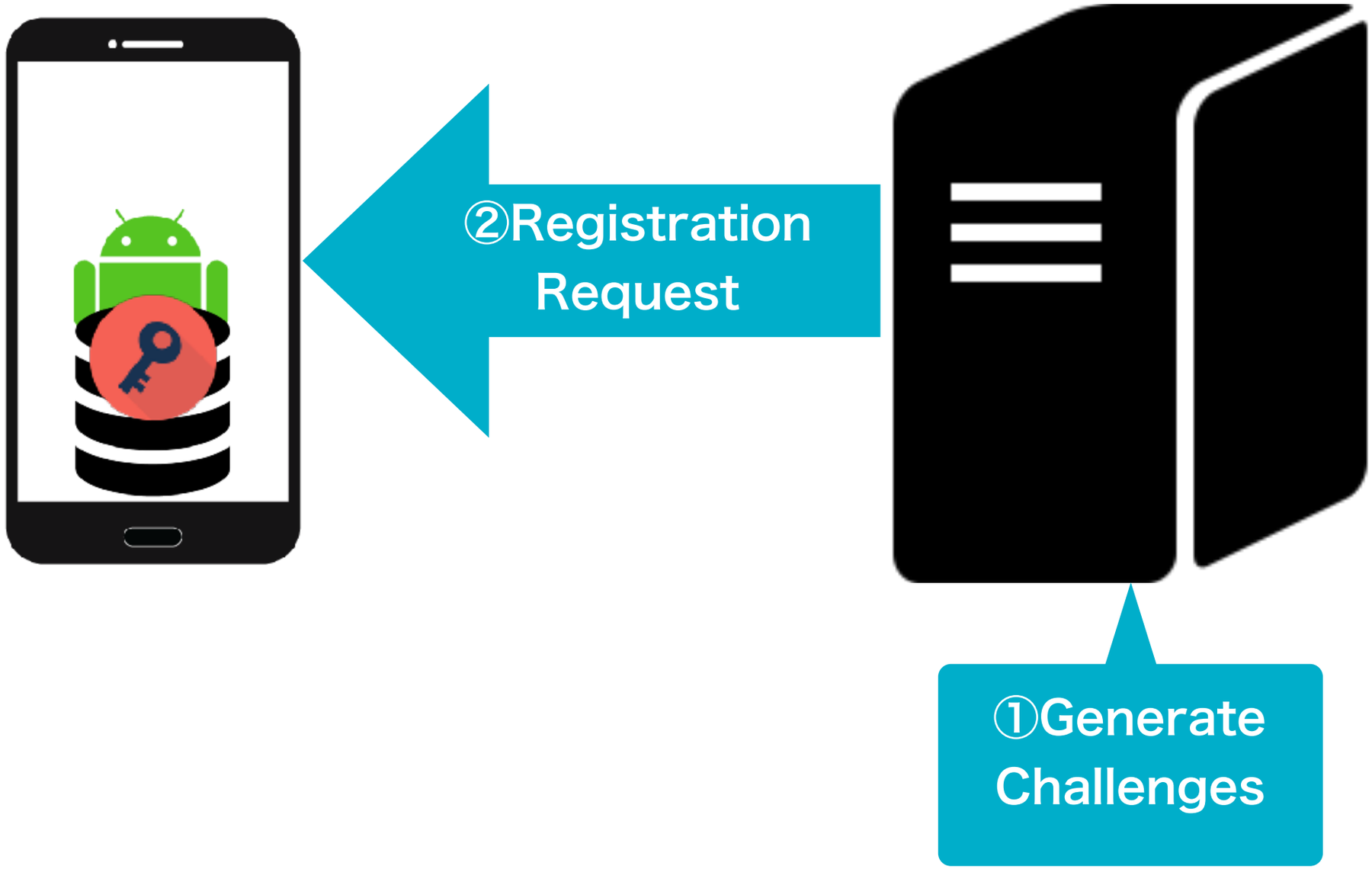


Key Attestation in Android

- From **Preinstalled Oreo**
  - with **Google Play** Installed
  - All the device **must be shipped** with Attestaion Key
- For **Preinstalled Nougat**
  - **some** Device has Attestation Key
  - Only **HW-backed KeyStore is ensured**

Registration Flow

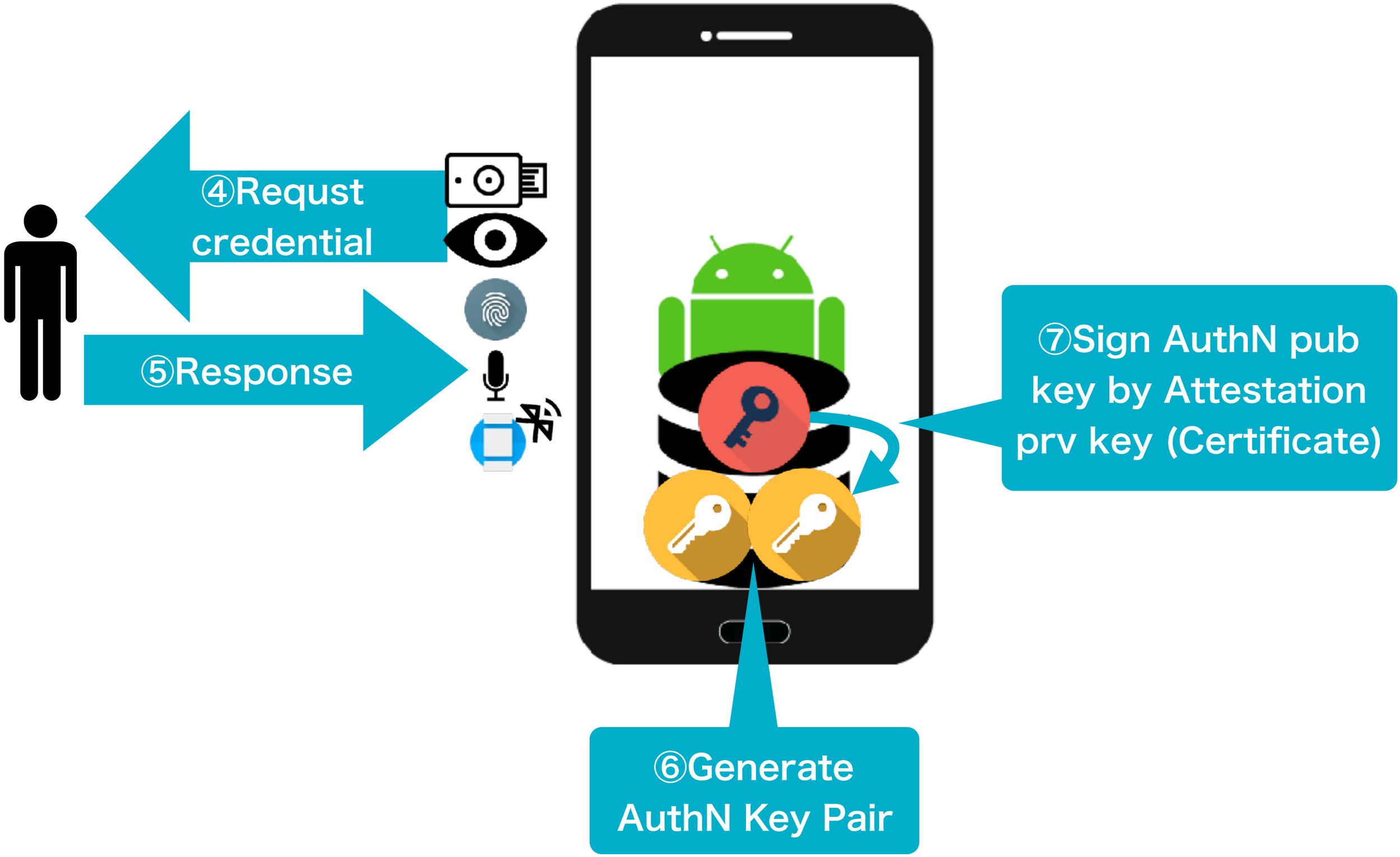
Registration Flow



Registration Request

```
[{
"header": {
    "upv": { "major": 1, ”minor": 1 },
    "op": "Reg", <-
    "appID": "https://hoge.com/SampleApp/uaf/facets",
    "serverData": "xxxxxxxxxxxxxxxx"
},
"challenge": “this_is_challenge”,
"username": “ken5scal”, <-
"policy": {
    "accepted": [ … ],
    "disallowd": [ … ]
}
```

Generating AuthN_Key Pair



Generating AuthN_Key Pair

```
val notBefore = Calendar.getInstance()
val notAfter = Calendar.getInstance()
notAfter.add(Calendar.YEAR, 10)

val kpg = KeyPairGenerator.getInstance(
  KeyProperties.KEY_ALGORITHM_EC,"AndroidKeyStore")
val builder = KeyGenParameterSpec.Builder(
  "authN_key", KeyProperties.PURPOSE_SIGN)
builder
  .setDigests(KeyProperties.DIGEST_SHA256)
  .setAlgorithmParameterSpec(
    ECGenParameterSpec("prime256v1"))
  .setUserAuthenticationRequired(true)
  .setCertificateSubject(
    X500Principal(String.format("CN=%s, OU=%s",
    "authN_key", applicationContext.packageName))
  .setKeyValidityStart(notBefore.time)
  .setKeyValidityStart(notAfter.time)
```

Generating AuthN_Key Pair

```
builder.setAttestationChallenge(finalHash)
kpg.initialize(builder.build())
val authNKeyPair = kpg.generateKeyPair()
```

Registration Response

Retrieving Certificate

```
val ks = KeyStore.getInstance(
  KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore")
ks.load(null)
val certificate = ks.getCertificate("authN_key")
val signedPubKey = Base64.encodeToString(
 certificate.encoded,
 Base64.NO_WRAP or Base64.NO_PADDING)
```

Registration Response


```
[{
"assertions": [{
  "assertion": "attestationInByteArray",
  "assertionScheme": "UAFV1TLV" }],
"fcParams": "finalHash",
"header": {
  "appID": "https://hoge.com/SampleApp/uaf/facets",
  "op": "Reg",
  "serverData": "xxxxxxxxxxxxxxxxx",
  "upv": { "major": 1, "minor": 1} }
}]
```

Registration Response

⑨Verify Cert by Attestation pub Key provided by FIDO Alliance

Revisiting: Protecting Private Key

- ✓ Is it **stored in safely manner**
- ✓ How to **verify storing method**
- ✗ Lost/Change Device

FIDO for Android

Summary

FIDO UAF for Android Summary

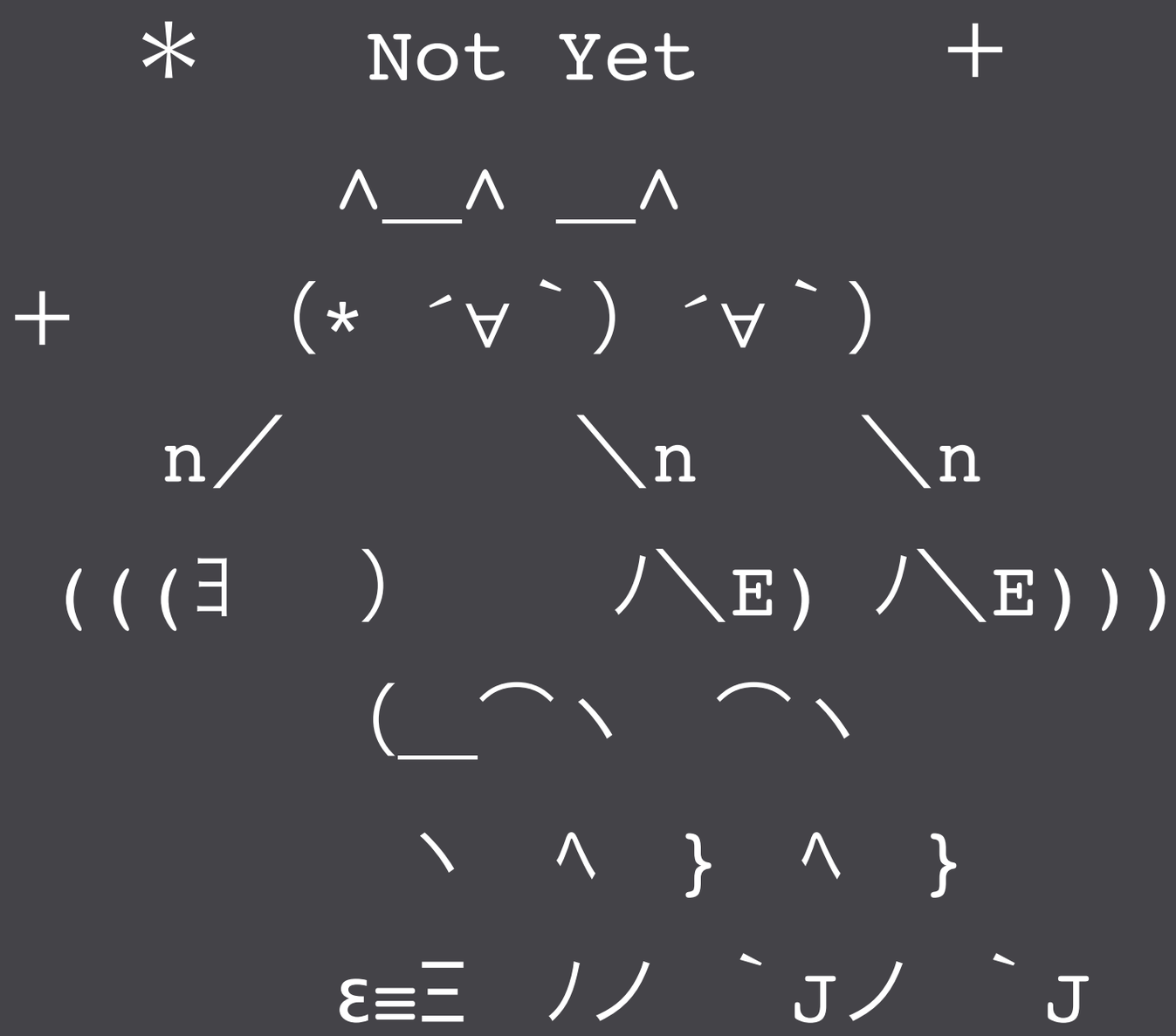
- **2FA by Default**
  - Use Authenticator to authN user locally
- Send signed verification over network (AuthN device remotely
- Protecting private key
  - **TEE**
  - **Key Attestation**


OREO


OAUTH
2
OAUTH


fido
TM
alliance

Is API Protected?


* Not Yet +
^__^ __^
+ (* ´∀`) ´∀`)
n/ \n \n
(((∃) 人E) 人E)))
(_⌒ヽ ⌒ヽ
ヽ ^ } ^ }
ε≡Ξ ノノ `Jノ `J

No such thing as 100% risk free

Keep Protecting

Thank You!