University of Piraeus

Department of Digital Systems

Postgraduate Program «Digital Systems Security»

Academic Year 2017-2018

(ΨΣ-ΑΦ-888) – MSc Dissertation

Mobile Connect Authentication with EAP-AKA

Bountakas Panagiotis
MTE1626, panampoyntakas@ssl-unpi.gr

Supervisor

Associate Professor Xenakis Christos

Piraeus, March 2018

This Thesis is dedicated to my family.

Special Thanks to Dr. Xenakis and Dr. Dadoyan for their guidance and to my fellow student Vaio Bolgoura for his help.

Contents

Table Of Figures

Abstract

To log into an application, users need quite often to have an account, i.e. usernames and passwords. This becomes a problem when the number of accounts keeps increasing. Mobile Connect [1] offers a simple and fast way for users to log into an application by matching the users to their mobile phones. In order to achieve that the user should be authenticated by their Mobile Network Operator (MNO). Until now this happens with various methods, i.e. SMS, Smartphone App Authenticator (SAA), SIM Applet, FIDO, USSD, which have either security or complexity issues. To address these drawbacks, we propose an alternative authentication method which utilize the EAP Authentication and Key Agreement (AKA) mechanism of the 3rd generation mobile networks, specified for Universal Mobile Telecommunication System (UMTS). We suggest this mechanism to be deployed as a SAA. What is more, for the EAP packets to be transferred through the internet, a method like HTTP Authentication with EAP or IPsec could be used.

Introduction

Nowadays as the digital economy expands, both individuals and businesses are looking for ways to easily interact online without compromising security. Modern world requires from us to have accounts everywhere. The majority of the accounts support a username-password combination. According an online survey, which surveyed more than 2,000 English-speaking adults, the average person has 27 discrete online logins [35]. 27 username-password combinations are way too much for one person to remember. Hence, this leads users to use the same password over and over again which is a major security issue, since if one of those accounts gets compromised and attackers retrieve the password then all the other accounts which have the same password will be compromised too. This makes the need of limiting the accounts that correspond in each person crucial. Technologies like Single Sign On (SSO), OAuth, OpenID Connect (OIDC) and Mobile Connect were created in order to mitigate this problem. SSO provides access control by combining many different and independent software systems and assigns a unique ID and password to each user to gain access to a connected system or systems. OAuth and OIDC let users log in an application or service by using their social media or email accounts, instead of creating new accounts. This made social media and email providers act as identity providers (IdPs). Moreover, Mobile Connect does the same but instead of using the social media or email providers in order to authenticate, it uses the mobile phone and by extension the identity in the Mobile Network Operator (MNO). All these methods have the same goal of helping people by using accounts they already have instead of creating new ones. As mentioned above, MNOs are responsible for users' authentication and use various methods to achieve it. Some of these methods have complicated implementation and others have security issues i.e. SMS. The whole procedure of the Mobile Connect protocol depends on the authentication process, so we believe that this process should be as secure as possible. Our approach aims at providing the most accurate and secure authentication, for that reason it uses the EAP AKA protocol to authenticate the subscriber; it is not a new protocol which means that its security has already been tested. The EAP AKA in order to be easily adopted should be implemented as a Smartphone App Authenticator (SAA) and for the EAP packets to be transferred through the internet we propose the use of the HTTP Authentication with EAP scheme or the IPsec in tunnel mode. Furthermore, MNOs are already familiar with this protocol hence the implementation will be plain sailing. Besides, users will not experience any difficulties using this authentication method, as it will be deployed as a smartphone application which will require the least possible interaction with the user. The solution described in this thesis is unique as the EAP AKA protocol has not been used with Mobile Connect yet.

Background Information

Mobile Connect

Mobile Connect is a secure universal login solution which allows users to login to websites and applications without the need of credentials, like usernames and passwords. It works by simply matching the user to their mobile phone. It offers a high level of security and no personal information is shared without permission, instead it uses Service Provider (SP) specific pseudonyms. This means that the SP does not know who the user is and cannot track a customer across different services. It is developed by the GSMA and its operator partners. GSMA represents the interests of mobile operators worldwide, unites nearly 800 of the world ‘s mobile operators, as well as more than 230 companies in the broader mobile ecosystem. The GSMA Mobile Identity program is focused on positioning operators as trusted providers of identity services to 3rd party SPs. As we mentioned above, having too many accounts is a major problem nowadays, so the need of password-less authentication methods is crucial. Mobile Connect is a very useful and convenient tool as it verifies user through their mobile phone, which everyone has always with them. At the time of writing (2018), it is currently being deployed by fifty-seven operators in thirty countries around the globe, including Asia, Europe, India and parts of the America and it is available to more than 2.5bn mobile users worldwide. By using the inherent security of the mobile devices carried by consumers, Mobile Connect can provide authentication at a number of security levels on any device i.e. PC, mobile or tablet. The key components of which it is composed are the API Exchange that helps the SP finds the MNO, in the MNO the Identity Gateway (ID GW) which connects with an authentication server and it is responsible for the authentication of the user and for the communication with the SP. It requires the service provider and the application to be registered in advance hence it adds an extra level of security by checking whether the service provider is legitimate. The registration process is pretty straightforward, all that is needed in order to use the service for the first time and includes agreeing in operator ‘s terms and conditions and confirming the mobile number. It helps online businesses increase their sales by lowering the likelihood of abandoned shopping carts, as it reduces the need of consumers to remember multiple username and passwords. The user is authenticated by the MNO using various methods. These methods will be explained briefly in the authentication methods chapter. The image below shows the protocol overview.

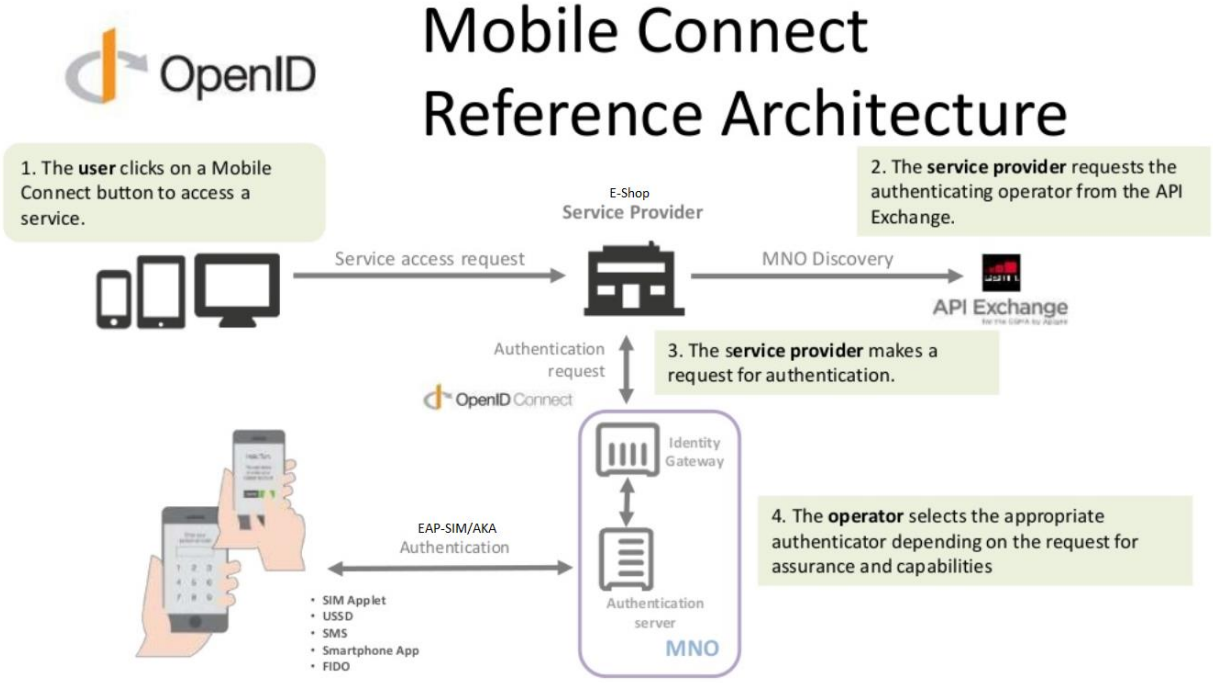


Figure 1: Mobile Connect Overview

1) Firstly, the user clicks on a Mobile Connect button to access a service e.g. As a checkout option.
2) The service provider requests the authenticating operator from the API Exchange. The API Exchange is used from Mobile Connect in order to discover the endpoint of the serving operator. Its use will be described in detail in the next section.
3) The service provider makes a request for authentication to the MNO. OIDC is used as the protocol interface between the service provider and the MNO. The service provider only connects to Mobile Connect ID GW of the service operator.
4) The operator authenticates the end user using one of the existing authentication methods i.e. SMS, SIM Applet, Smartphone App, etc. The ID GW of the MNO is responsible for the authentication of the end user. This is the entity which selects the appropriate authentication method. It also implements the OIDC protocol and generates the ID tokens.

API Exchange

In Mobile Connect users are initially asked to enter their mobile numbers. The mobile number is not shared with the website or app. Instead, it is used by the API Exchange, which is provided by GSMA, to discover the user ‘s mobile operator, in order to be able to make a request directly to the operator. The API Exchange is a central component to manage discovery and enable federation across the Identity Provider (IDP). It exposes a simple REST based Discovery API ultimately returning “Discovered” resources in JSON object. It uses Mobile Country Code (MCC)/Mobile Network Code (MNC) or network IP address to discover the serving operator. In case the client initiates the discovery off net or off channel, the API presents the user a form to

select the home operator. The operator should provide to the API Exchange the MCC/MNC, IP address range, OIDC endpoints and certificate details during the onboarding process. About the operator implementation of the API Exchange, it will be used as the recommended federation mechanism for Mobile Connect. It ensures that Service Providers only need to connect to one operator of a federation to get access to customers of all connected operators.

The figure below describes the discovery process of the MNO.

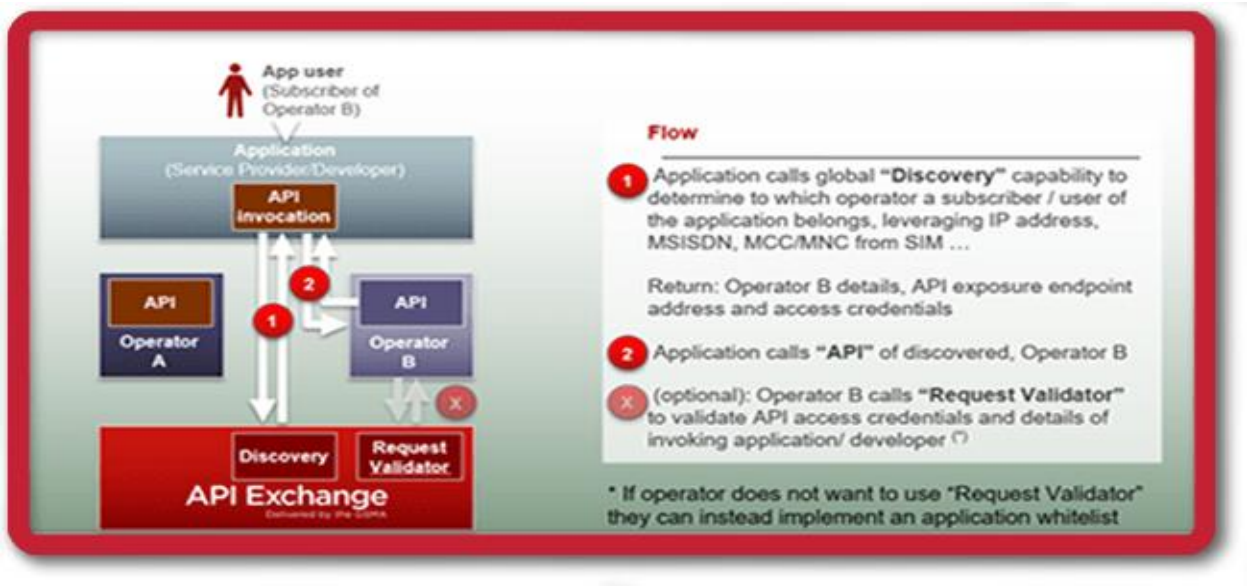


Figure 2: API EXCHANGE

Mobile Connect Foundations

Mobile Connect is based on OICD protocol and OIDC is based on the OAuth protocol, hence these three protocols are interconnected, so in the below pages we explain briefly the OAuth and OIDC protocols.

OAuth 2.0

OAuth 2.0 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service, such as social media, email and HTTP service providers in general. It delegates the user authentication to the service that hosts the user account and authorizing 3rd party application to access the user account. It defines four roles: Resource Owner i.e. the user, Client i.e. the application, Resource Server and Authorization Server. An abstract protocol flow is shown in the above figure.

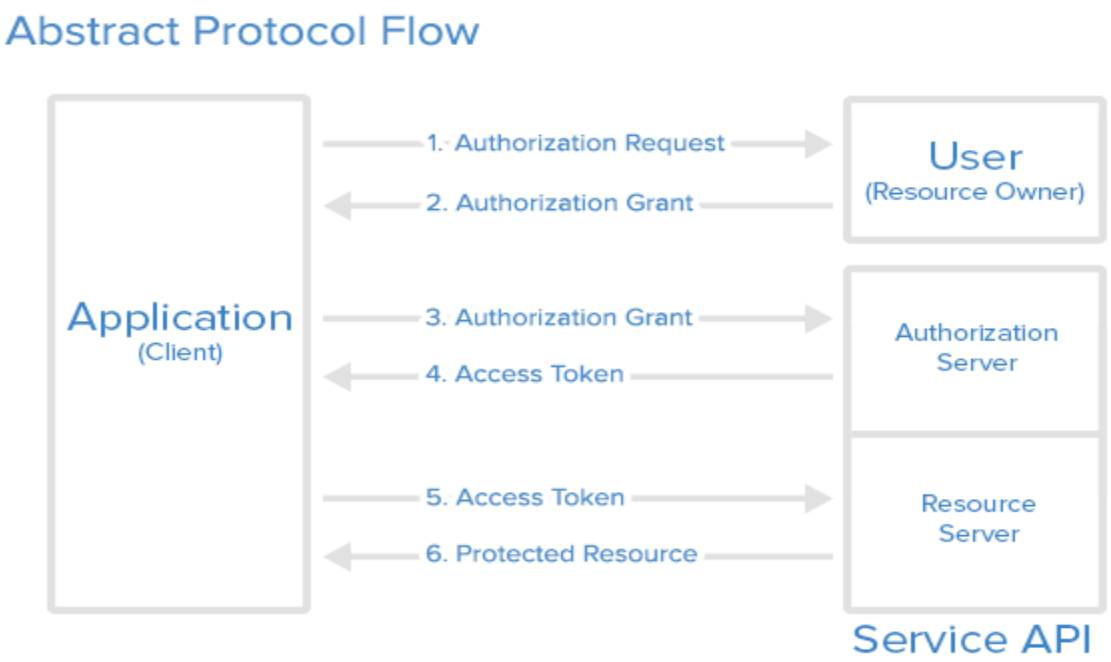


Figure 3: OAUTH2.0 OVERVIEW

1. The *application* requests authorization to access service resources from the *user*
2. If the *user* authorized the request, the *application* receives an authorization grant
3. The *application* requests an access token from the *authorization server* (API) by presenting authentication of its own identity, and the authorization grant
4. If the application identity is authenticated and the authorization grant is valid, the *authorization server* (API) issues an access token to the application. Authorization is complete.
5. The *application* requests the resource from the *resource server* (API) and presents the access token for authentication
6. If the access token is valid, the *resource server* (API) serves the resource to the *application*

The real flow of this process will differ depending on the authorization grant type in use.

The use of OAuth within an application implies that it should be registered with the service. This is done through a registration form in the service's website. In this form the following information should be provided: application Name, application website, redirect URI or callback URI.

The redirect URI is where the service will redirect the user after they authorize (or deny) the application, and therefore the part of the application that will handle authorization codes or access tokens.

Once the application is registered, the service will issue "client credentials" in the form of a client identifier and a client secret. The Client ID is a publicly exposed string that is used by the service API to identify the application, and is also used to build authorization URLs that are

presented to users. The Client Secret is used to authenticate the identity of the application to the service API when the application requests to access a user's account, and must be kept private between the application and the API.

Authorization Grant

The grant type flows determine how the Access Token are returned to the Client. The different grant types are:

- Authorization Code: Usually used with server-side application
- Implicit: Used with mobile apps or applications that runs on the user ‘s device.
- Resource Owner Password Credentials: Used with trusted applications, such as those owned by the service itself.
- Client Credentials: Used with applications API access.

Authorization Code Flow

The most commonly used grant type is the authorization code grant, because it is optimized for *server-side applications*, where source code is not publicly exposed, and *Client Secret* confidentiality can be maintained. This is a redirection-based flow, which means that the application must be capable of interacting with the *user-agent* (i.e. the user's web browser) and receiving API authorization codes that are routed through the user-agent. The authorization code flow is described below:

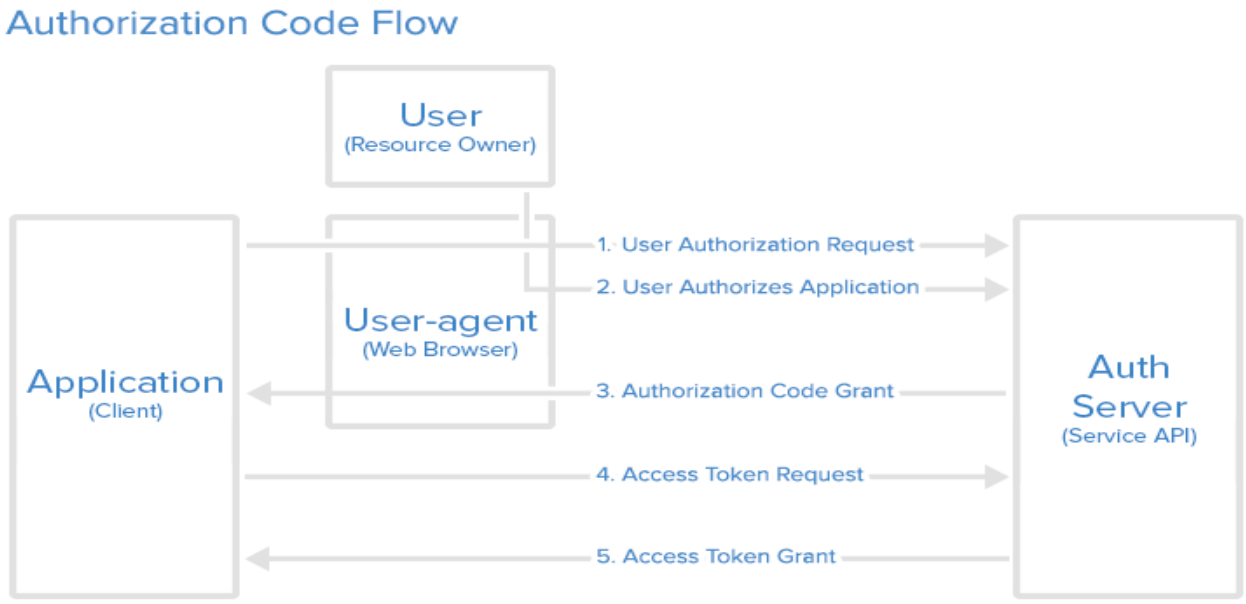


Figure 4: AUTHORIZATION CODE FLOW

1) The user is given an authorization code link which have the following components:

- https://www.example.com/ v1/oauth/authorize: the API authorization endpoint
- client_id=id: the application's client ID
- redirect_uri=callback_URI: where the service redirects the user-agent after an authorization code is granted
- response_type=code: specifies that the authorization code grant type is used
- scope=read: specifies the level of access that the application is requesting
- Full example: https://www.oauth.example.com/v1/oauth/authorize?response_type=code&client_id=id&redirect_uri=callback_URL&scope=read

2) The user clicks the link, firstly they must log in to the service, in order to authenticate their identity (unless they are already logged in). Then they will be prompted by the service to authorize or deny the application to access their account.
3) If the user chooses the "authorize" option, the service redirects the user-agent to the application redirect URI, which was specified during the client registration, along with an authorization *code*. Example: https://application.com/callback?code=AUTHORIZATION_CODE
4) The application requests an access token from the API, by passing the authorization code along with authentication details, including the client secret, to the API token endpoint.
5) For a valid authorization the API will send a response containing the access token to the application. It may use the token to access the user's account via the service API until the token expires or is revoked.

Implicit

The implicit grant type usually used for mobile apps and web applications, where the client secret confidentiality is not guaranteed. This grant type is also a redirection-based flow but the access token is given to the user-agent to forward to the application, hence it may be exposed to the user and other applications on the user's device. Also, this flow does not authenticate the identity of the application, and relies on the redirect URI to serve this purpose. The full process using implicit grant type is discussed below:

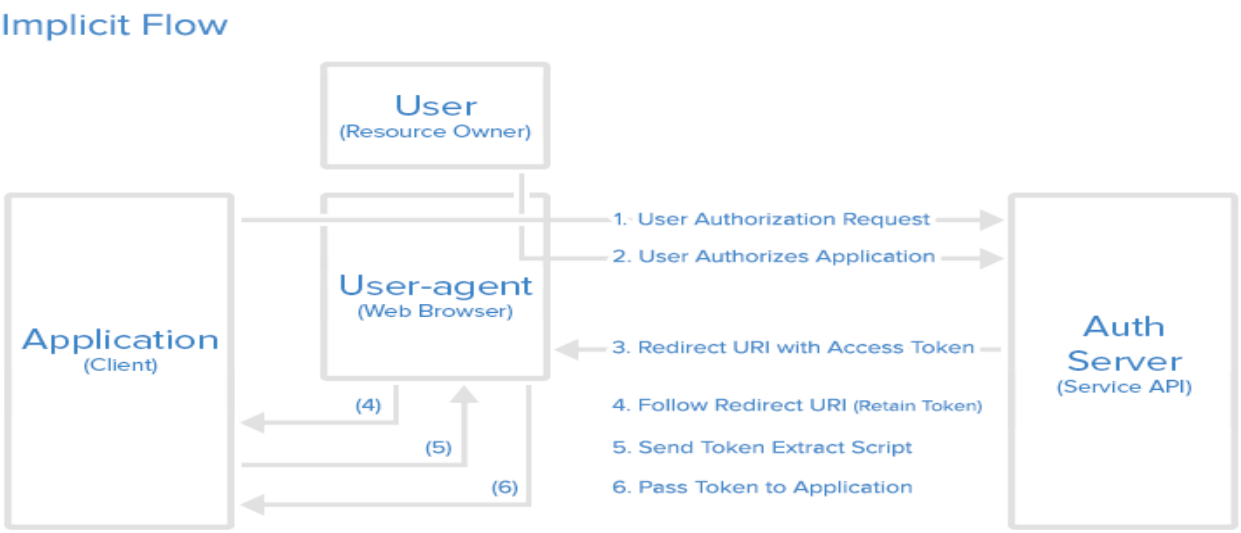


Figure 5: IMPLICIT FLOW

1) With the implicit grant type, the user is presented with an authorization link, that requests a token from the API. This link looks just like the authorization code link, except it is requesting a token instead of a code.
2) This step is the same with the authorization code. The user clicks the link, firstly they must log in to the service, in order to authenticate their identity (unless they are already logged in). Then they will be prompted by the service to authorize or deny the application access to their account.
3) If the user chooses the "authorize" option, the service redirects the user-agent to the application redirect URI, and includes a URI fragment containing the access token. It would look like this: https://application.com/callback#token=ACCESS_TOKEN
4) The user-agent follows the redirect URI but retains the access token.
5) The application returns a webpage that contains a script that can extract the access token from the full redirect URI that the user-agent has retained.
6) The provided script is executed by the user-agent and passes the extracted access token to the application. It may use the token to access the user's account via the service API until the token expires or is revoked.

Resource Owner Password Credentials

The user provides his service credentials (username and password) directly to the application, which uses the credentials to obtain an access token from the service. This grant type should only be enabled on the authorization server, if other flows are not viable. Also, it should only be used if the application is trusted by the user.

After the user shares his credentials with the application, the application will then request an access token from the authorization server. This request might look like: https://www.oauth.example.com/token?grant_type=password&username=USERNAME&password=PASSWORD&client_id=CLIENT_ID

If the user credentials check out, the authorization server returns an access token to the application and the application is authorized.

Client Credentials

The client credentials grant type provides a way to the application of accessing its own service account. This might be useful if the application wants to update its registered description or redirect URI, or access other data stored in its service account via the API.

The application requests an access token by sending its credentials, its client ID and client secret, to the authorization server. This request might look like: https://www.oauth.example.com/token?grant_type=client_credentials&client_id=CLIENT_ID&client_secret=CLIENT_SECRET

If the application credentials check out, the authorization server returns an access token to the application and the application is authorized.

OpenID Connect - Mobile Connect Profile

The Mobile Connect is based on the OpenID Foundation OpenID Connect standard. The OpenID Foundation formed in 2007 and is a non-profit international standardization organization of individuals and companies committed to enabling, promoting and protecting OpenID technologies. OIDC is published in 2014 and it is an identity layer on top of the OAuth 2.0 protocol. It enables clients to verify the identity of the user based on the authentication performed by an authorization server. In OIDC the roles differ a little from OAuth. The Resource Owner is called End-User, the Client is called Relying Party (RP), the Resource and Authorization server are called OpenID Provider (OP). The OIDC protocol supports three flows for obtaining the ID tokens:

- Authorization code flow: This is the most commonly used flow, intended for traditional web apps as well as native/mobile apps. Involves an initial browser redirection to/from the OP for a user authentication and consent and then a second back channel request is used to retrieve the ID token.
- Implicit flow: Usually used by the browser based apps that do not have a backend and the ID token is received directly.
- Hybrid flow: It is a combination of the two previous flows. This is the most rarely used which allows the app front-end and back-end to receive tokens separately from one another.

As it mentions before, the OIDC reuses the OAuth 2.0 protocol and parameters and extends it to introduce an Identity Layer through the following additions:

- Along with the access token, an ID token is returned, which is basically a JSON Web Token with identity claims (user information).

- A UserInfo endpoint introduced, which returns basic profile attributes against the access token.

ID Token

Client applications receive the user‘s identity with an ID Token, which is the primary extension that OIDC makes to OAuth 2.0 to enable End-Users authentication. The ID Token is basically a JSON Web Token (JWT). JWT is an open, industry standard (RFC 7519) method that defines a compact and self-contained way for securely transmitting information between two parties as a JSON Object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA. JWT are useful for authentication, as when the user is logged in, each subsequent request will include the JWT, allowing the user to access routes, services, and resources that are permitted with that token. Single Sign On is a feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains. Also it is useful for information exchange, since it can be signed and you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.

Anatomy of a JSON Web Token (JWT)

A JWT consists of three base 64 encoded strings separated by “.“ and it looks like aaaaaaa.bbbbbbbbbbb.cccccccc .

This three parts are:

- Header
- Payload
- Signature

Header

The header carries two parts, declaring the type (which is JWT) and the hashing algorithm to use (for example HMAC SHA256). So the header before encoded to base 64 look like:

{

“typ”: “JWT”,

“alg”: “HS256”

}

And in base 64 format: **eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9**

Payload

The payload will carry the JWT claims. This is where the information that we want to transmit and other information about the token. There are multiple claims that we can provide. This includes registered claim names, public claim names and private claim names.

Registered claims

Claims that are not mandatory whose names are reserved for us. These include:

- iss: The issuer of the token
- sub: The subject of the token
- aud: The audience of the token
- exp: This will probably be the registered claim most often used. This will define the expiration in NumericDate value. The expiration MUST be after the current date/time.
- nbf: Defines the time before which the JWT MUST NOT be accepted for processing
- iat: The time the JWT was issued. Can be used to determine the age of the JWT
- jti: Unique identifier for the JWT. Can be used to prevent the JWT from being replayed. This is helpful for a one time use token.

Public claims:

These are the claims that we create ourselves like username and other important information.

Private claims:

A producer and a consumer may agree to use claim names that are private.

The payload before the base 64 encode look like:

```
{
  "iss": "scotch.io",
  "exp": 1300819380,
  "name": "Chris Sevilleja",
  "admin": true
}
```

And in base 64 format:
eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOjEzMDA4MTkzODAsIm5hbWUiOiJDaHJpcyBTZXZpbGxlamEiLCJhZG1pbiI6dHJ1ZX0

Signature

The third part of JWT is the signature which is made up of a hash of the header, payload and secret. The secret is the signature held by the server. This helps the server verify existing tokens and sign new ones. The signature is constructed as it shown below:

var encodedString = base64UrlEncode(header) + "." + base64UrlEncode(payload);

HMACSHA256(encodedString, 'secret');

And is base 64 format:
03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773

The full JWT is:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzY290Y2guaW8iLCJleHAiOjEzMDA4MTkzODAsIm5hbWUiOiJDaHJpcyBTZXZpbGxlamEiLCJhZG1pbiI6dHJ1ZX0.03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773

OpenID Connect Implementation in Mobile Connect

The OIDC is used for the communication between the SP and the MNOs ID GW. Hence, a specific profile of OIDC that should be implemented by the MNOs is needed.

The image below illustrates an abstract protocol flow. The actual flow may have additional steps depending on the authorization grant model used, e.g. the Authorization Code grant model need an additional step, from the implicit flow, to get the access token using the authorization code.

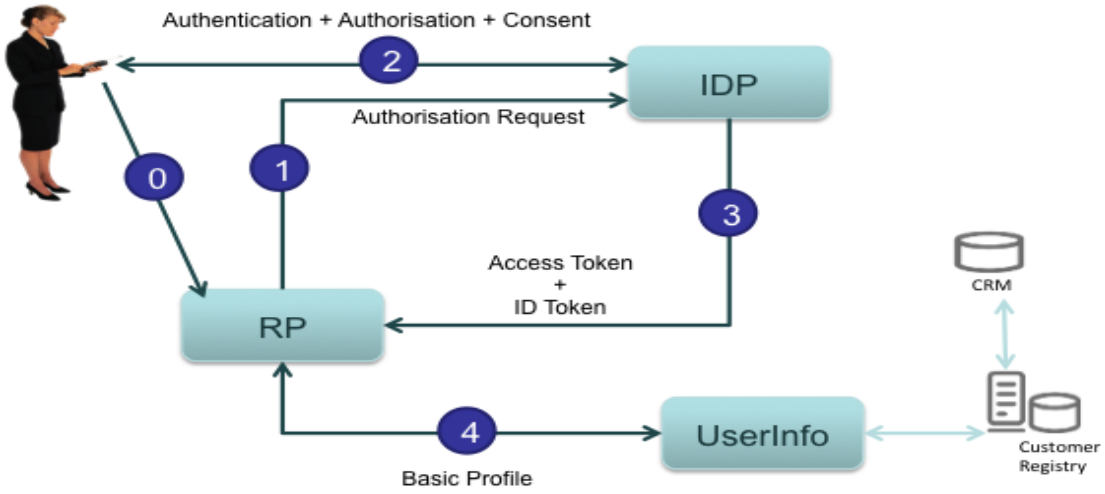


Figure 6: OPEN ID CONNECT IMPLEMENTATION IN MOBILE CONNECT

The description of the above diagram is:

0) The user is using the service from the SP/RP and the SP/RP needs to authenticate the user.
1) The SP/RP prepares and sends the OIDC authorization request to the ID GW of the MNO.

2) The ID GW selects the appropriate authenticator and authenticates the user.
3) The ID GW returns to the SP/RP, the response that is depending on the grant type used. The SP/RP gets the anonymized identifier and the authentication context.
4) The SP/RP may call the UserInfo end point at the MNO to get the basic attributes by passing the access token.

Registration/Provisioning of Relying Party

The RP, in the OIDC implementation for Mobile Connect, is the SP application. The application developers should first register their client applications with the IDP/MNO/OP. This step is required to improve user information security. The registration process involves at least the following:

- The RP must provide one or more redirect_uri to be used in order to respond back for Authorization Requests through redirect.
- The RP will receive client_id, client_secret to be used for request authentication and authorization. However, the client_secret is not used in the Implicit flow.

Authorization Code Flow

The grant type that is recommended for the Mobile Connect profile is the Authorization Code flow. It is determined by the response_type parameter in the Authorisation Request initiated by the SP/RP towards the IDP/MNO/OP (ID GW). The Authorization Code flow obtains the Access Token and the ID Token using a two-step process as described in the figure below.

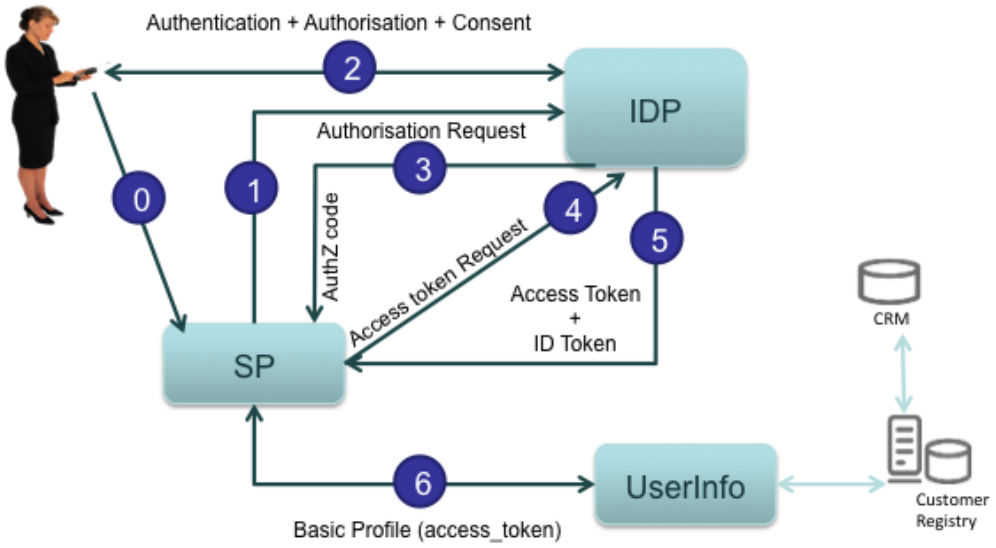


Figure 7: AUTHORIZATION CODE FLOW

Some steps are similar to the one of the abstract flow.

0) The user is using the service from the SP/RP and the SP/RP needs to authenticate the user.
1) The SP/RP prepares and sends the OIDC authorization request to the ID GW of the MNO.
2) The ID GW selects the appropriate authenticator and authenticates the user.
3) The IDP returns the Authorisation Code to the SP/RP.
4) The SP/RP sends the token request to the token end-point at the ID GW, passing the Authorization code.
5) The ID GW validates the Authorisation code and returns the Access Token along with the ID Token – containing the authorization context. The SP/RP gets the anonymized identifier and the authentication context.
6) The SP/RP may call the UserInfo end point at the MNO to get the basic attributes by passing the access token.

Authorization Request & Response Example

RP sends the Authorization Request to the ID GW. The request is sent using the HTTPS protocol via GET or POST method. A sample request using the GET method is presented below:

```
GET /authorize?
response_type=code&
client_id=s6BhdRkqt3
&redirect_uri=https%3A%2F%2Fclient.mid.org
&scope=openid
&state=af0ifjsldkj
&nonce=n-0S6_WzA2Mj
HTTP/1.1
Host: mid.example.com
Accept: application/json
```

The ID GW authenticates user, gets user consent and returns the "code" to the RP/SP. The code is returned to the URI value specified in the redirect_uri, response parameters are included as query parameters encoded using application/x-www-form-urlencoded.

Sample response:

```
HTTP/1.1 302 Found
Location: https://client.mid.org?code=SplxlOBeZQQYbYS6WxSbIA
&state=af0ifjsldkj
```

Authentication methods

The act of confirming the truth of an attribute of a single piece of data claimed true by an entity is called authentication. When this has to do with humans authenticating on machines, it is the

verification of credentials required for confirmation of a user's authenticity. Traditional authentication methods consist of an ID and password combination, and through the years, alternative authentication methods - like smart cards and biometrics - were developed. In addition, more authentication factors are added to improve the security of communications. Main authentication factors are based on something I know e.g. a password, something I have e.g. a sim card, something I am e.g. fingerprint. Mobile connect is based on something I have, the mobile phone and by extend the sim card. This means that it is less vulnerable to a scalable compromise/attack than the password based mechanisms. Also, it supports two factor authentication methods when needed. The various authentication methods that can be used in Mobile Connect are presented in this chapter.

SMS and URL-based Authenticator

Mobile Connect uses an SMS One Time Password (OTP) based approach in order to authenticate users. More specifically the ID GW generates an OTP in the form of session ID and adds it in a URL, which is sent in the authentication device of the user as an SMS. The user opens the SMS and clicks the URL provided in the message. The pros are that it is simple and reuses mobile operator assets, but comes with the drawbacks that is not very secure, as the SMS can be intercepted by attackers and other apps on the device.

The full authentication process is described below.

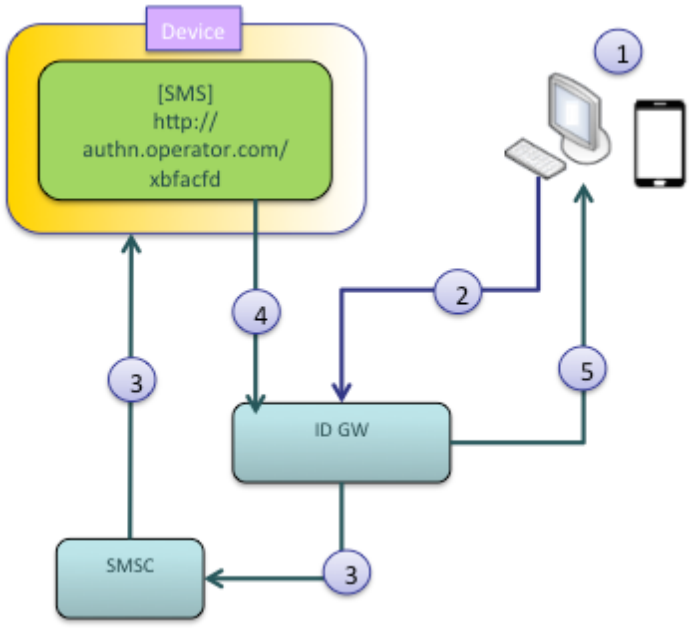


Figure 8: SMS BASED AUTHENTICATION

1) The user accesses the service provider's web page via PC, mobile phone, tablet or any internet device over any type of internet access.

2) The user clicks on the mobile connect link.
3) The SP app sends an OIDC request to the MNO ID GW. An OTP in the form of a session ID/Transaction ID is added in the URL passed in the SMS
4) The ID GW generates an OTP in the form of a session ID/Transaction ID and adds it in a URL, which is then sent to the authentication device of the user as an SMS. The user receives the SMS message. The user may need to unlock the mobile phone at this time. The user opens the SMS message and clicks the URL provided in the message.
5) The URL points to the ID GW
6) The OTP in the URL is verified. The ID GW sends the authentication response to the SP.

SIM Applet Authenticator

SIM Application Toolkit (SIM Applet) is a standard of the Global System for Mobile Communications (GSM) that enables the SIM card to initiate actions which can be used for various value-added services, like voicemail box, mobile advertising, etc. For the authentication of the user within Mobile Connect the SIM Applet pops up a user interface with a *Click OK* experience or a *PIN* for higher level of security. It is easy to use and the PIN is stored is SIM and never transmitted, but it has complex architecture, it has a dependency on the SIM and it does not work over non mobile operator network e.g. Wi-Fi. The authentication with SIM Applet is shown in the figure below.

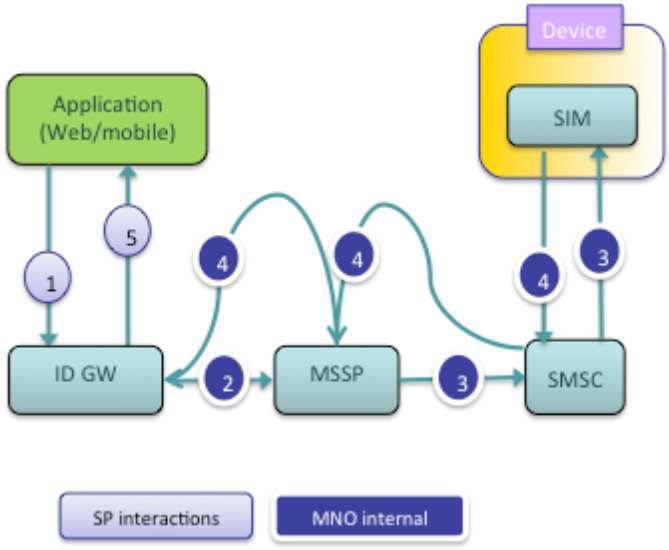


Figure 9: SIM APPLET AUTHENTICATION

1) The application (desktop, tablet or mobile phone) sends an OIDC request.
2) The ID GW sends a request to the Mobile Signature Service Platform (MSSP) using the ETSI 102 204 SOAP API.
3) The MSSP sends a Class 2 SMS through the SMSC.
4) The SIM applet pops up a user interface (UI) to present a *Click OK* experience or *PIN* experience.
5) The applet validates the PIN.

6) The applet sends the authentication response with an encrypted MO SMS through the SMSC, MSSP to the ID GW.
7) The ID GW returns back with the OIDC response.

Fast Identity Online (FIDO) Authenticator

The FIDO Alliance has defined a framework that enables a local device authentication via biometrics to be used for online approvals. In order to implement it within Mobile Connect the solution consists of,

- An authentication client which is installed on the user ‘s device.
- An authentication server that integrates with the MNO (or SP)
- The Universal Authentication Protocol (UAF) that allows the client and the server to communicate.

The following figure illustrates how the FIDO framework could be integrated into the Mobile Connect architecture as another authentication mechanism.

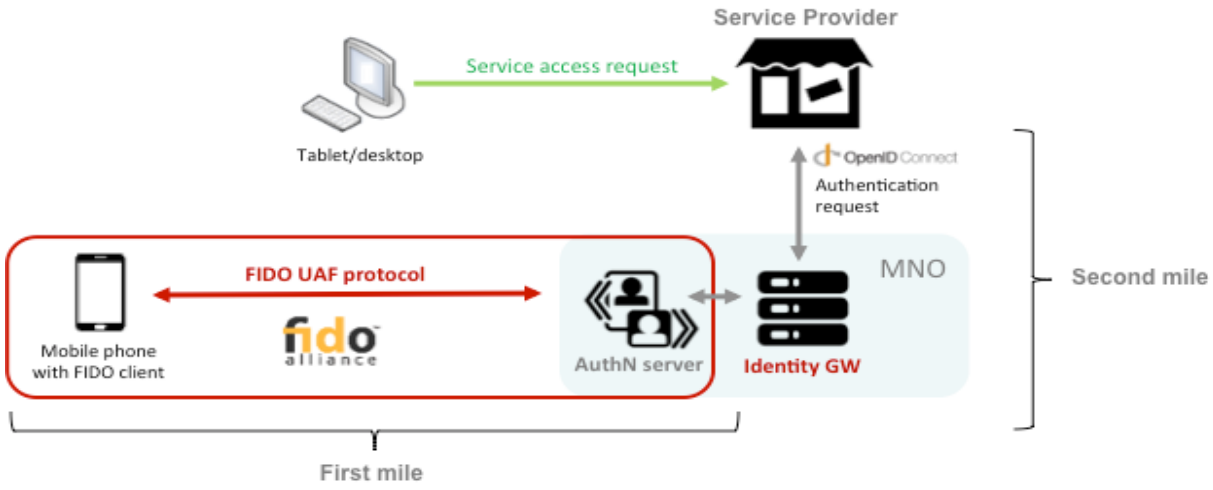


Figure 10: FIDO AUTHENTICATION

FIDO Authenticator Using SIM as the Secure Element

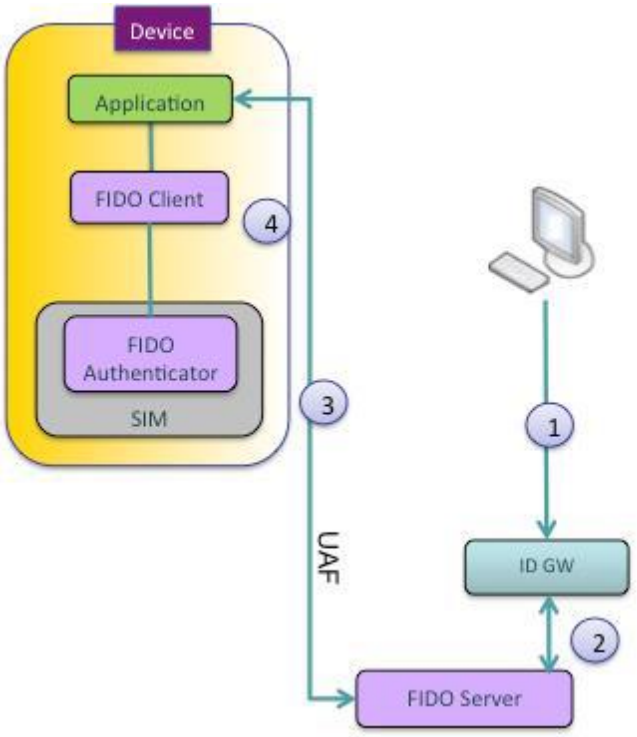


Figure 11: FIDO USING SIM AUTHENTICATION

The above figure illustrates the FIDO authenticator implemented in the SIM card (as an applet) and FIDO client interacts with the FIDO authenticator using Open Mobile APIs.

1) The SP app sends an OIDC request to the ID GW.
2) The ID GW sends an authentication request to the FIDO server.
3) The FIDO server interacts with the application on the device using UAF protocol.
4) The application uses the FIDO stack in the device to perform the authentication and sends the authentication response back to the FIDO server, which is then sent back to the SP.

USSD-based Authenticator

Unstructured Supplementary Service Data is a communication network used by GSM cellular telephones to communicate with the MNO's computers. It is usually used for prepaid callback service, mobile money services, menu based information services, etc. The ID GW of the MNO interacts with the USSD GW to send the USSD message to the user's device. The device pops up a USSD menu to type the PIN or whatever is asked for. This mechanism also supports the *Click OK* and the *PIN* authentication modes. The pros that come with this method are that it uses the mobile operator assets, it works in roaming conditions across devices and it does not depend on a data channel. However, it has minimal user experience and most importantly it does not work with Long Term Evolution (LTE).

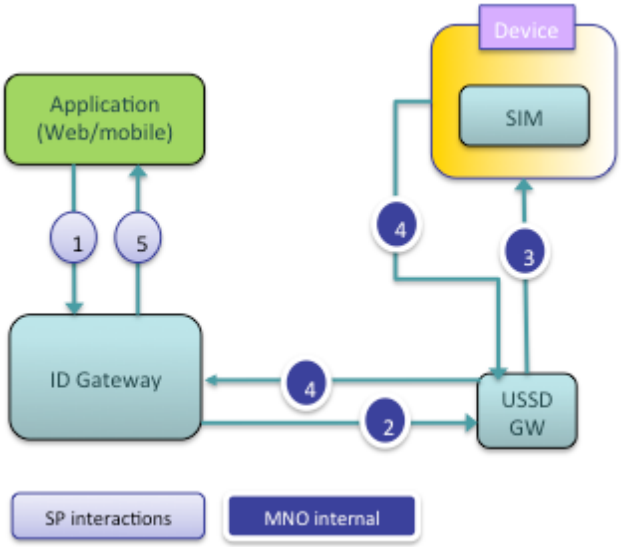


Figure 12: USSD BASED AUTHENTICATION

The above image shows the authentication process with a USSD-based authenticator.

1) The application (desktop, tablet or mobile phone) calls an OIDC Authorization towards the mobile operator ID GW to authenticate the user.
2) The mobile operator ID GW interacts with the USSD GW to send USSD messages.
3) The USSD GW sends a message to the device.
4) The device pops up a USSD menu to type the PIN. For seamless login, this can simply type 1 to authorize and 0 to cancel, if 1FA is sufficient.
5) The USSD message is sent back to the ID GW, through the USSD GW.
6) The ID GW service responds to the application.

Smartphone App Authenticator

The Smartphone App Authenticator (SAA) is a mobile application which uses the richness of the user experience and device features available in the smartphone to enhance the experience when using this kind of authenticator. In order to use the SAA, the user should download the application and install it in the mobile phone. In order for the app to be able to authenticate the user it has a setup process which is transparent in the user. After this process the app will be ready to authenticate the user when it is needed. During the authentication procedure the app prompts the user to *Click OK* or to enter the *PIN*. This authentication method offers a rich user experience, it uses the device capabilities and it can be extended to use biometrics. However, it may have app cloning issues, especially if the device is compromised. The setup and the authentication process is described more analytical below.

Setup mode prepares the app for the authentication mode.

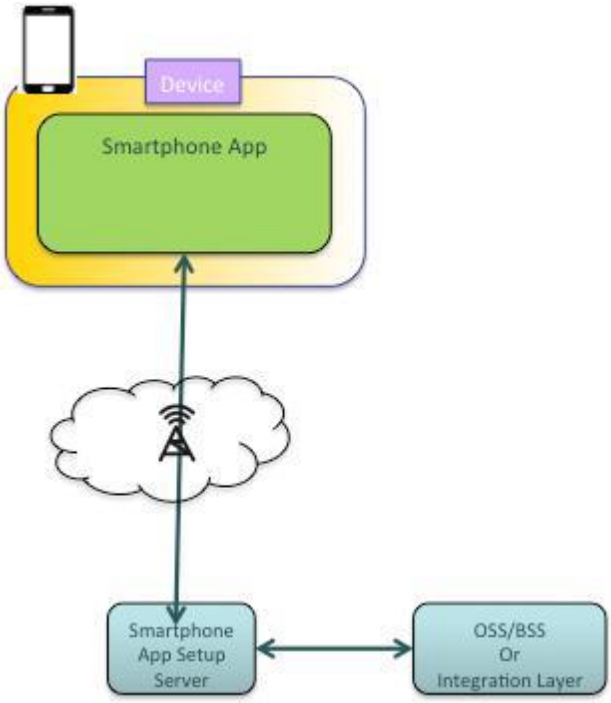


Figure 13: SAA SETUP MODE

1) Smartphone app connects using the mobile network.
2) App reads the IMSI and IMEI using the native SDK.
3) App sends setup request to the setup server through the mobile network (including IMSI and IMEI).
4) The mobile operator core network adds the authenticated MSISDN into the HTTP header.
5) The setup server asks the OSS/BSS through internal APIs to get the associated IMSI and IMEI of the attached device for the MSISDN.
6) The setup server validates the setup request from the app by comparing IMSI, IMEI from the 2 sources (app ‘s request and network).
7) Setup server generates a token based on the IMSI, IMEI and MSISDN.
8) The token is sent in the response to the app.
9) The app securely stores the token.
10) The token is used for signing the interactions between the smartphone app and the authentication system.

Authentication mode is used for authenticating the end user.

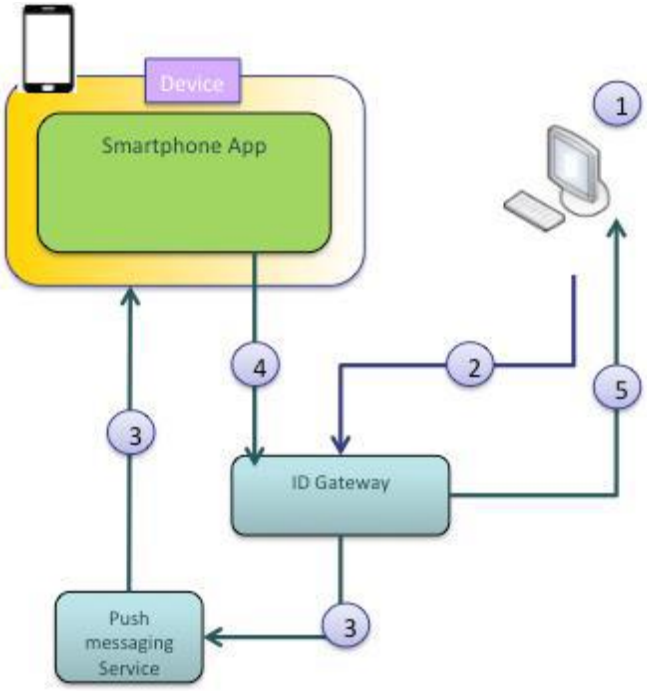


Figure 14: SAA AUTHENTICATION MODE

1) User accesses service provider ‘s web page (to identify the operator API Exchange is used).
2) SP server sends an authentication request using the OIDC protocol in the operator ‘s ID GW.
3) The ID GW policy engine decides to use and route to the app authenticator. The ID GW sends a message to the app through the push messaging service and the app prompts the user to click OK or enter a PIN.
4) The encrypted response is sent back to the ID GW using HTTPs, adding a signature using the token received during the setup phase. The ID GW decrypts and validates the message, a signature validation is performed at the ID GW to establish the validity of the application.
5) The OIDC response is sent to the SP.

EAP based Authentication

It has already been mentioned, that our proposed method uses the EAP AKA protocol for the authentication of the user within Mobile Connect. In order for the reader to be able to understand our methodology and consequently the use case scenario presented in a later chapter, we describe below the protocols which are based on EAP and used in our suggested methodology.

EAP is an authentication framework which supports multiple authentication methods. It runs directly over data link layers such as Point to Point Protocol (PPP) or IEEE 802, thus, it does not require an IP address. The advantage of the EAP architecture is its flexibility, as it is used to select a specific authentication mechanism, typically after the authenticator requests more

information in order to determine the specific authentication method to be used. EAP permits the use of a backend authentication server, which may implement some or all the authentication methods, with the authenticator acting as a pass-through for some or all methods and peers. EAP supports more than forty different authentication methods, some of them are: EAP - Transport Layer Security (TLS), EAP - MD5, EAP - Tunneled Transport Layer Security (TTLS), EAP – Subscriber Identity Module (SIM), EAP – Authentication and Key Agreement (AKA), etc.

EAP is not a wire protocol; it only defines message formats. Hence, each protocol that uses EAP defines a way to encapsulate the EAP messages within the protocol ‘s messages. The most known protocols that encapsulate EAP are:

- IEEE 802.1x: EAP over IEEE 802 or EAP over LAN (EAPOL) is defined in IEEE 802.1x. When EAP is invoked by an 802.1x enabled Network Access Server (NAS), modern EAP methods can provide a secure mechanism and negotiate a secure private key between the client and NAS that can then be used for a wireless encryption session.
- Protected Extensible Authentication Protocol (PEAP): PEAP is a protocol that encapsulates EAP within a TLS tunnel. It aims to correct deficiencies in EAP; EAP assumed a protected communication channel, such as that provided by physical security, so the facilities for protection of the EAP conversation were not provided. It was jointly developed by Microsoft, Cisco Systems and RSA Security.
- Point to Point (PPP): EAP was originally an authentication extension for the PPP and was created as an alternative to the Challenge Handshake Authentication Protocol (CHAP) and the Password Authentication Protocol (PAP). Eventually the above two protocols were incorporated into EAP.
- RADIUS and Diameter: These two are Authentication Authorization and Accounting (AAA) protocols, which encapsulate EAP messages. They are often used by NAS devices to facilitate IEEE 802.1x by forwarding EAP packets between IEEE 802.1x endpoints and AAA servers.

The EAP packets are defined in a binary format and their contents highly depend on the authentication scheme that is used each time. The EAP packet format is describing in the figure below. The fields are transmitted from left to right.

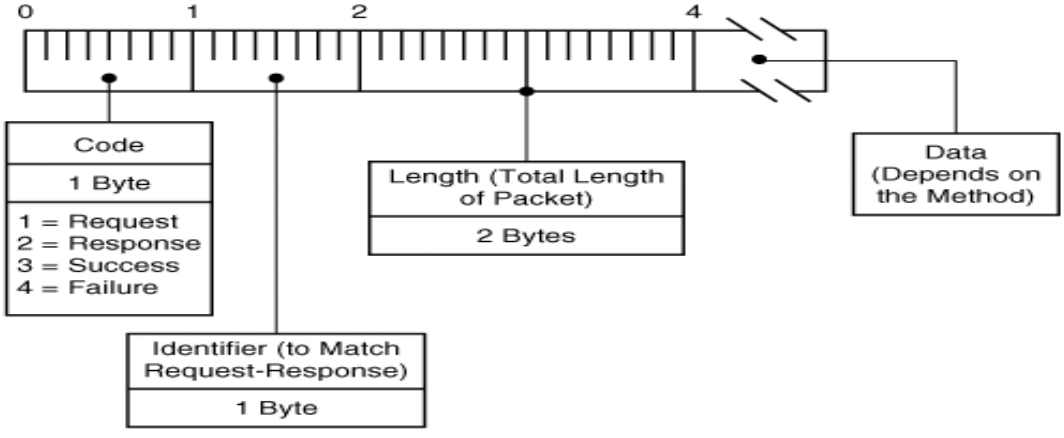


Figure 15: EAP PACKET FORMAT

EAP AKA

As mentioned before, our proposed methodology suggests the implementation of EAP AKA within Mobile Connect. This chapter describes briefly the features of EAP AKA, the protocol overview will be presented as a use case scenario of the user authentication by the MNO in a later chapter. EAP AKA defines the EAP mechanism for authentication and session key distribution that uses the Authentication and Key Agreement (AKA) mechanism. AKA is used in the 3rd generation mobile networks Universal Mobile Telecommunications System (UMTS). It is based on challenge-response mechanism and symmetric cryptography and typically runs in a SIM, which is a UMTS SIM (USIM). The EAP AKA is applied in several new applications, including the following:

- The use of the 3rd generation mobile network authentication infrastructure in the context of wireless LANs.
- Relying on AKA and the existing infrastructure in a seamless way with any other technology that can use EAP.
- The use of AKA also as secure PPP authentication method in devices that already contain an identity module.

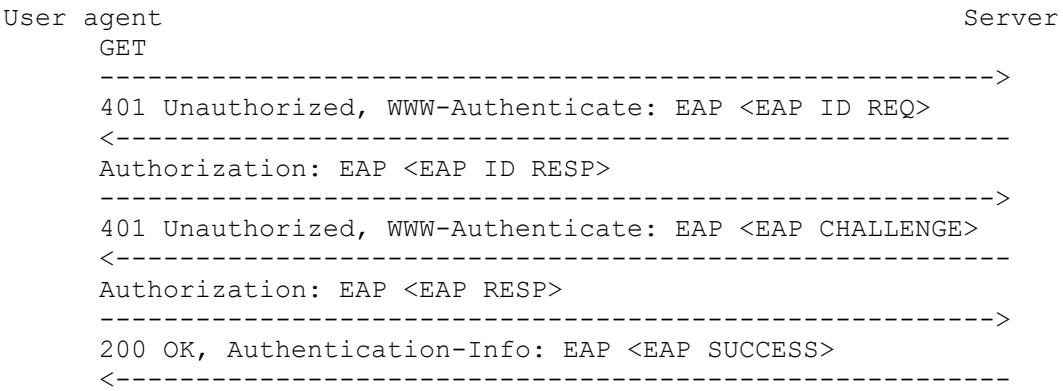
In our approach the EAP AKA protocol is used for the authentication of a user which is using the Mobile Connect service. In the operator, it should be implemented in the ID GW which is responsible for the authentication of the users. The EAP AKA mechanism should be deployed as a smartphone application in order to be easily used by the users.

EAP AKA Communication Through Public Internet

The EAP AKA messages in order to be transferred through the internet, from the user to the MNO, typically it should be encapsulated within RADIUS or Diameter protocol, but this would add several drawbacks in our approach, so we came up with alternative methods to transfer the EAP packets through the internet without the need of a RADIUS or a Diameter server to be implemented. We found two methods that could be used for our purpose. These two methods are: the encapsulation of the EAP messages within HTTP Authentication headers or the use of IPsec protocol in tunnel mode. What is more, in this chapter we describe briefly the above methods.

HTTP Authentication with EAP

The HTTP Authentication framework [22] includes two authentication schemes: Basic and Digest. In the Basic scheme, the client authenticates itself with a user-ID and a password for each realm. This scheme is perceived as insecure since the user credentials are transmitted in a clear text format. The Digest scheme is based on cryptographic hashes and is more secure than the Basic scheme, but is limited to the use of passwords. The HTTP Authentication scheme supplements HTTP authentication with EAP functionality. The EAP packets encoded as base64 format and delivered within HTTP Authentication headers. All the relevant information about the authentication scheme is included in EAP packets. The content of these packets depends on the chosen EAP authentication method. A typical message sequence is presented in the below diagram.



In order to make EAP as an independent HTTP authentication scheme, three new header types were defined for the HTTP authentication framework. These headers are the WWW-Authenticate Response Header, Authorization Request Header and Authentication-Info Response Header.

The WWW-Authenticate Response Header using EAP authentication would look like:

WWW-Authenticate: eap realm="BollyWorld@example.com", eap-p="QWxh4ZGRpb2jpvcGVuNlctZQ=="

The “BollyWorld” is the string assigned by the server to identify the protection space of the Request-URI at server “example.com”.

The Authorization Request Header using EAP authentication would look like:
Authorization: Eap realm="BollyWorld@example.com",
eap-p="QWxhZGRpbjpvcGVuIHNlc2FtZQ=="
The rules for handling user identifiers, password, challenges and so on, are defined in EAP protocol [27].

The Authentication-Info Response Header is used by the server to communicate information back to the client. This can be either the successful authentication in the response, or the continuation of the EAP mechanism. The Authentication-Info Response header using EAP authentication would look like:
Authentication-Info: QWxhZGRpbjpvcGVuIHNlc2FtZQ==

IP Security

Internet Protocol Security (IPsec) is a framework, for a set of protocols, that authenticates and encrypts the packets of data sent over a network. It uses cryptographic security services to protect communications over IP networks. While some other internet security services in widespread use, such as the Secure Shell (SSH) and the Transport Layer Security (TLS), which operate in application and in transport layer respectively; IPsec can automatically secure applications at the IP layer. The IPsec suite is an open standard, which uses the following protocols to perform various functions:

- Authentication Header (AH): It provides connectionless integrity and guarantees the data origin by authenticating the IP packets. Optionally a sequence number can protect the IPsec packets against replay attacks. AH operates directly on top of IP.
- Encapsulating Security Payload (ESP): It provides origin authenticity through source authentication, data integrity through hash functions and confidentiality through encryption for IP packets. In transport mode the ESP does not provide integrity and authentication for the entire IP packet. Like AH the ESP operates directly over IP.
- Security Associations (SA): Is typically a one-way relationship between a sender and a receiver, where the communicating parties establish shared security attributes such as algorithms and keys. Before the data exchange starts the sender and the receiver agree on which algorithm is used to encrypt the IP packet. Also, the algorithm for authentication is agreed before the data transfer takes place. The SAs of IPsec are established using the Internet Security Association and Key Management Protocol (ISAKMP).

The IPsec supports two modes of operations:

- Transport mode: In this mode, only the payload of the IP packet is usually encrypted or authenticated. The IP header is neither modified nor encrypted, so the routing is intact. However, when the AH is used, the IP addresses cannot be modified by the Network Address Translation (NAT), as this always invalidates the hash value.
- Tunnel mode: In this mode, the entire IP packet is encrypted and authenticated. It is then encapsulated into a new IP packet with a new IP header. It is usually used to create Virtual Private Networks (VPNs).

IPsec tunnel with EAP AKA

The VoWiFi service in order to be used requires an IPsec tunnel to be created between the mobile device and an evolved Packet Data Gateway (ePDG). The ePDG is located in the MNO and its function is to terminate millions of IPsec tunnels. The use of IPsec allows the operator to route customer traffic across untrusted Wi-Fi and internet connections, which offers the same level of data integrity and protection as LTE using SIM credentials. The below figure illustrates how the IPsec is used in the VoWiFi.

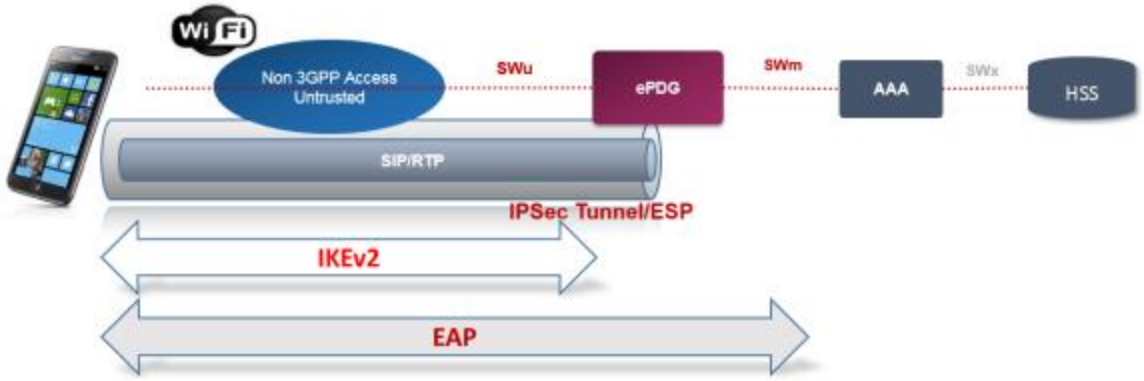


Figure 16: IPsec in VoWiFi

Besides the use of HTTP Authentication with EAP, for the transmission of the EAP packets from the mobile device, through the internet in the MNOs ID GW, there is another method, which is based on the philosophy of the VoWiFi. Moreover, the IPsec tunnel can use the Internet Key Exchange version 2 (IKEv2) for encryption and then the EAP AKA for authentication. IKEv2 is a component of IPsec, it is used for performing mutual authentication, establishing and maintaining SAs for ESP and AH. Among others, it supports EAP authentication. Thus, an IPsec tunnel could be created between the smartphone application, which will implement the EAP AKA mechanism, and the ID GW of the MNO, in order the EAP AKA communication to be happened.

Motivation

We live in a time that the demand on every technological aspect is high. Cell phones models and computers that were considered standard and were widely used about ten years ago, would be mostly deemed unacceptable today. That fact does not apply only to the hardware, but also the software and generally the way services are delivered to consumers. The number of businesses that offer their services online has dramatically increased over the last years, as the newest generations embrace technology easier and older users also adapt, even though it takes a bigger effort for them to make online purchases. Having all that in mind, we came up with a solution that will benefit all the parties mentioned above.

Using Mobile Connect with EAP-AKA to complete a purchase benefits the end user in many ways. First and foremost, it buys users time and spares them the dreadful process of filling forms at checkout. According to *Barilliance* [24] a very high percentage of users abandon their shop cart at that point.

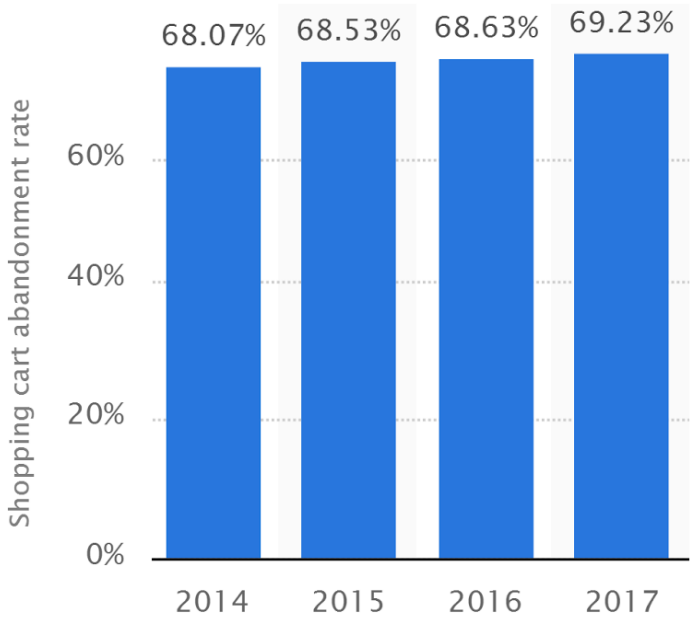


Figure 17: SHOPPING CART ABANDONMENT RATE

Baymard [23] mentions that two of the top three reasons this happens is basically because forms need to filled in order to complete the process.

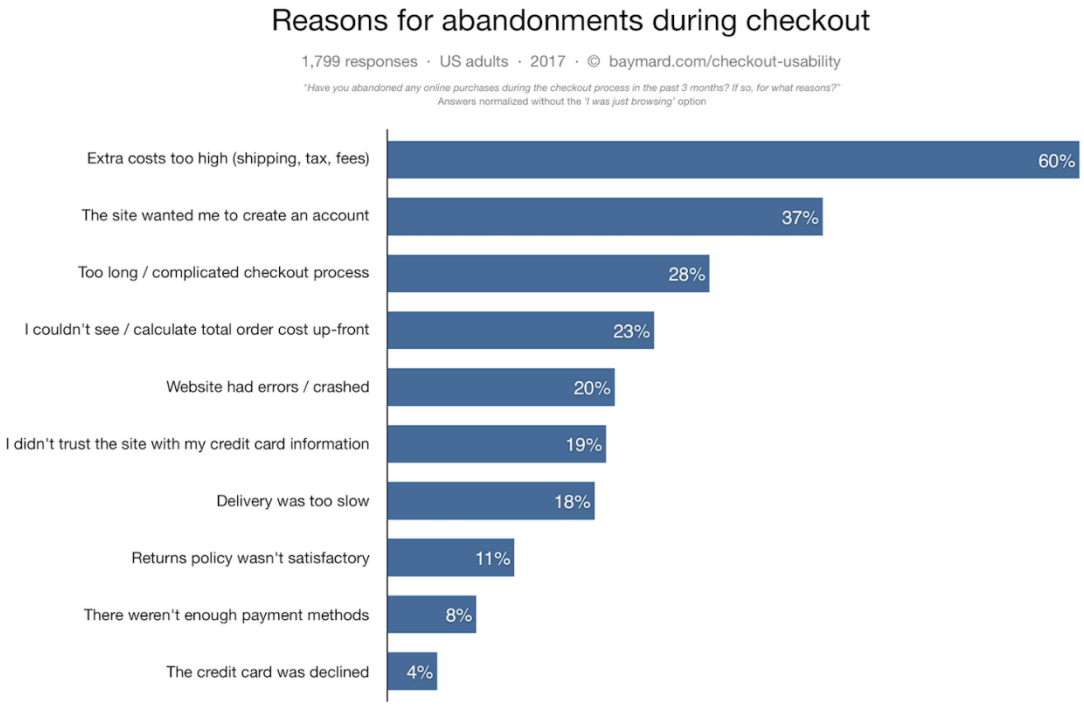


Figure 18: REASONS FOR ABANDONMENT DURING CHECKOUT

Note: subjects could choose more than one option

Completing forms can be time consuming and definitely discourages users from finalizing their shopping, thus we have unsatisfied consumers and loss of profits for the service providers.

Memorizing credentials, and the relatively low security they provide, is another problem many people are trying to solve nowadays. The Gartner Group suggests that about 70% of help-desk request are about forgotten credentials. Using password managers is kind of ironic security-wise, and biometrics are not a solution just by themselves. SSO solutions with Facebook, Google, etc, are starting to spread, but why should the user disclose any more personal information (like address and credit card info) to these companies? MNOs in most occasions already have the information needed to complete a purchase, as well as the infrastructure to authenticate the customer; all that remains is the user to give his consent to the MNO to share the information needed by the service provider. This is also a more secure method, as there are no passwords to be stolen and EAP-AKA authentication cannot be replicated by a malicious third party.

Many companies rely on the online sales they make and the implementation of our solution could mean an increase in their revenue and, looking at the big picture, boost the economy. Customers (especially new ones) will not leave at checkout anymore because a form is too time consuming and complicated, but instead they will finalize the purchase.

Contribution

Mobile Connect supports various authentication methods for the authentication of the users. One of those methods, as we have already seen in “Authentication Methods”, is the SAA. Our proposed method combines some characteristics of a SAA, as we suggest the deployment of EAP AKA as a smartphone application in order to be easily adopted by users. The proposed methodology is unique, since this authentication method is not used by Mobile Connect. Our approach combines already developed and tested technologies in the best way possible; EAP-AKA is already used in various systems to authenticate users via the 3G sim card from their devices, whilst Mobile Connect allows us to take advantage of this EAP method in applications. The fact that the solution proposed in this thesis is based on the coupling of others that already exist, means that the implementation is going to be neither difficult, nor particularly expensive. The MNOs are ready to implement the EAP AKA as an authentication method within Mobile Connect, since they have all the knowledge in order to do that. Moreover, the EAP AKA is mostly implemented for authentication in WLAN environments; not only is this use of EAP AKA quite different, but also it may lead to be used by other environments where a user should use their mobile phone in order to authenticate. What is more, we suggest two methodologies for the transmission of the EAP packets through the internet, HTTP Authentication with EAP and IPsec tunnel mode. On the one hand, there are not known implementations of HTTP Authentication with EAP AKA, hence this may open the road for the establishment of a new authentication scheme. On the other hand, the IPsec in combination with EAP AKA is used by VoWiFi services in order to authenticate the subscriber, but our proposed use of this method is quite different as it could be used in order to transfer the EAP AKA packets through the public internet and authenticate the user within Mobile Connect.

Analysis as a Use Case

The Scenario

Mobile Connect contributes in building security and privacy into digital services. It is a global and federated solution for mobile phone-based authentication, authorization, identity and attribute services. It can deliver appropriate security across many different use cases, including payment approval and public sector identification, without compromising user convenience or reach. Moreover, it helps online businesses increase their sales by lowering the likelihood of abandoned shopping carts, as it reduces the need of consumers to remember multiple username and passwords. For the above reason we choose the payment approval as a scenario to present our approach. Not only does this help us to see if our solution could be used as real use case scenario, but also it helps the reader to deeply understand our method.

More specifically as a use case scenario we consider the case in which a user wants to buy a book from an e-shop bookstore that is called ebookstore.gr. In our approach the EAP AKA Authentication mechanism should be deployed within a smartphone application, in order to be easily adopted by users. Also, the smartphone application should implement the HTTP Authentication with EAP AKA or the IPsec in tunnel mode with EAP AKA authentication, in order to transmit the EAP packets through the public internet in the ID GW of the MNO. This application should be fully compliant with GSMA Smartphone App Authenticator specifications described in "Smartphone App Authenticator" [2]. Also the app should offer a variety of lock settings, like pin or biometrics for providing an additional level of security; in order to prevent an unauthorized user from using it.

For our use case scenario, we consider that the user had already downloaded and installed the application in his/her mobile phone. In the first launch, the app should prompt the user to configure the lock settings. After that the app will be ready to use. We also consider that ebookstore.gr had already implemented the Mobile Connect service. Furthermore, the device that the user use to access the e-shop does not affect our scenario as soon as she/he has the mobile phone in the checkout time.

The entities which constitute the scenario are:

- The ebookstore.gr, which has the role of the SP.
- The user, that wants to make the buy.
- The user 's mobile phone, that has the EAP AKA application.
- The API Exchange, which is a part of the Mobile Connect Implementation.
- The MNO which serves the user.

The scenario focused on the authentication of the user which performed by the MNO, hence the authentication procedure was presented with all the details, but the other functions of the Mobile Connect were discussed briefly in order for the whole concept to be understandable for the reader.

Mobile Connect with HTTP EAP AKA Authentication

The user found the book that he/she wants to buy and continues with the checkout process. In order to checkout, the user has three options:

- Create Account,
- Guest Check Out,
- Mobile Connect.

An schematic overview of the whole procedure is shown below.

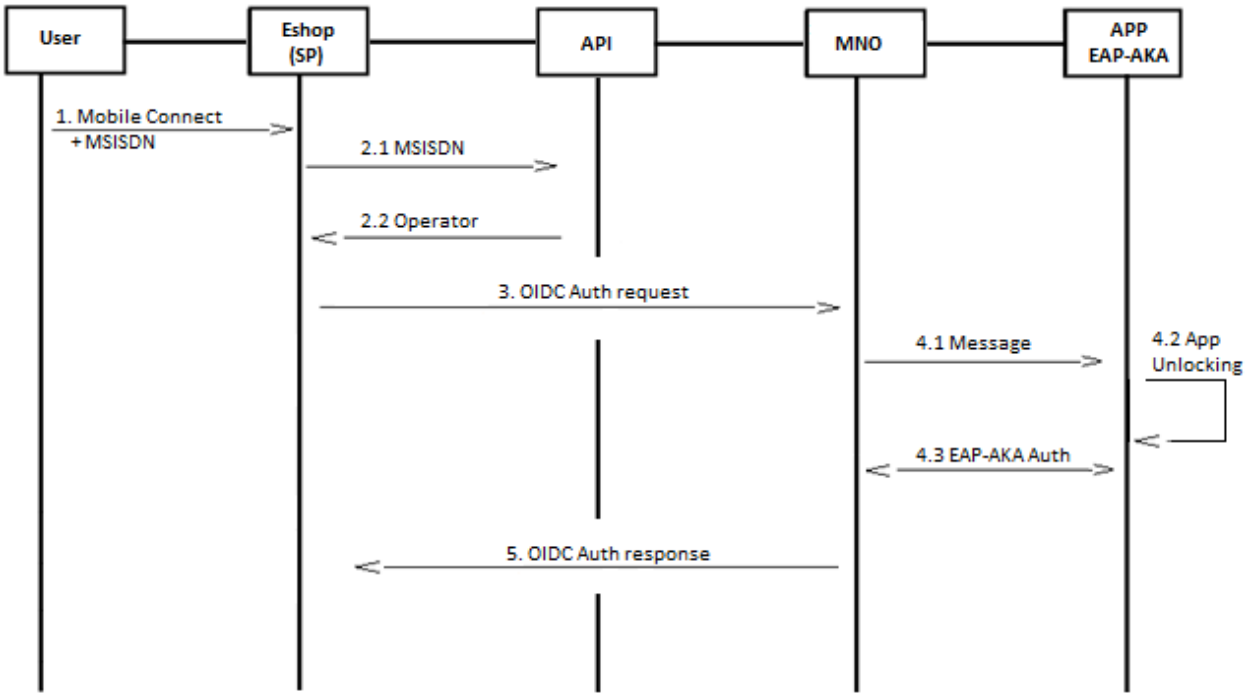


Figure 19: MOBILE CONNECT WITH EAP AKA AUTHENTICATION FLOWS

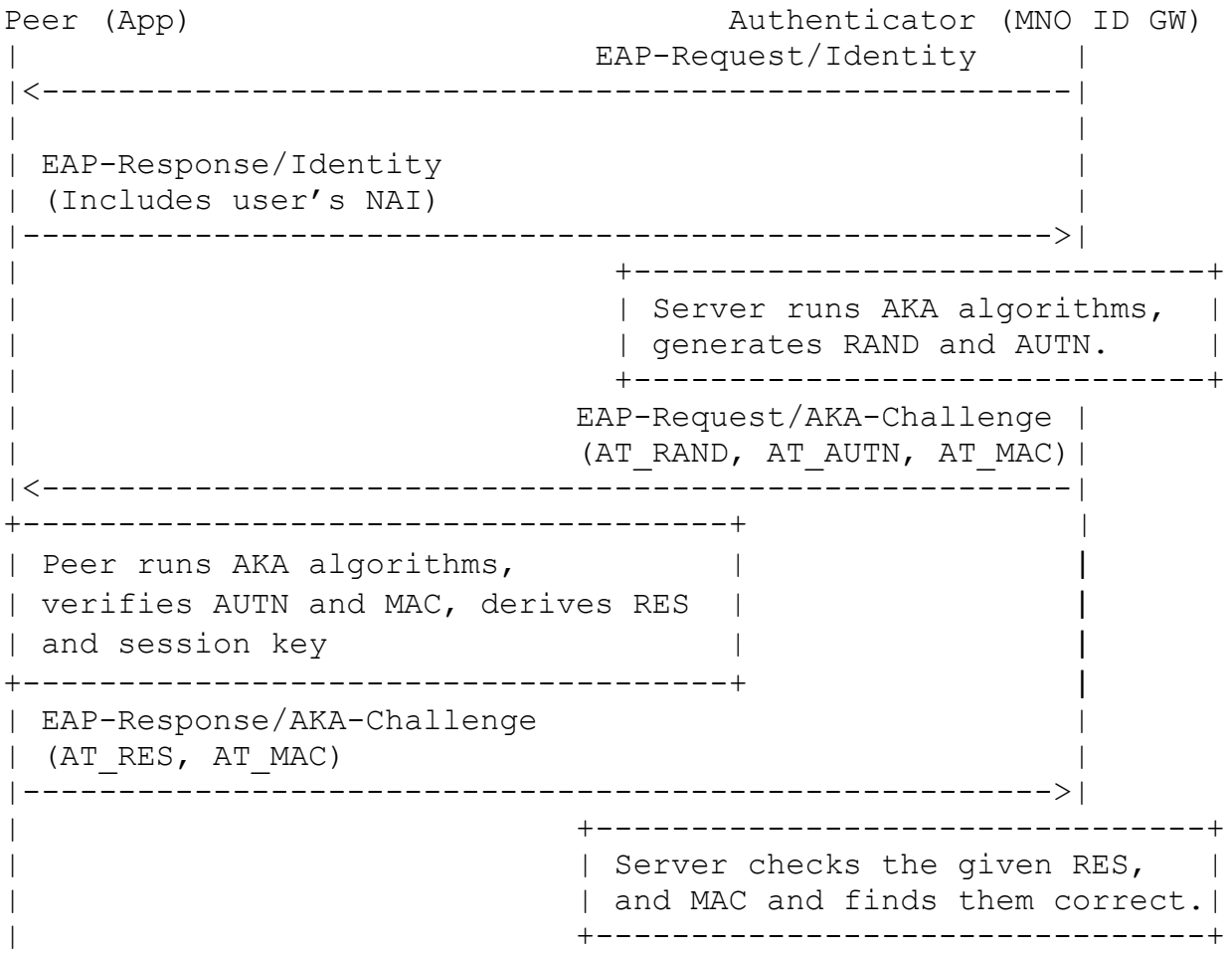
The user has no account in the ebookstore.gr and does not want to create one, neither wants to use the guest check out option as a big form should be filled. So the user chooses to use the Mobile Connect service and clicks the Mobile Connect button. The following steps are taking place in order to authenticate:

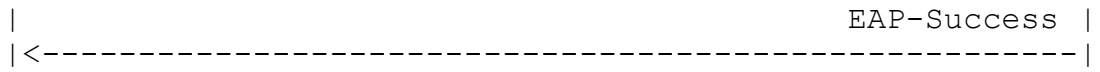
1 - At first a pop up window opens and the user is prompted to enter his/her mobile phone number. The privacy of the user is protected as the mobile number is not shared with the e-shop.

2 - The service provider (e-shop) requests the authenticating operator from the API Exchange, step 2.1[11]. The API Exchange as we mention before is a central component to manage discovery and enable federation across the IDPs. It is used by Mobile Connect in order to discover the endpoint of the serving operator. The serving operator is discovered and sent back on the SP as it shown in the step 2.2.

3 - The SP now knows the serving operator of the user and makes a request for authentication to the specific MNO. OIDC is used as the protocol interface between the service provider and the MNO. The service provider only connects to Mobile Connect Identity Gateway (ID GW) of the MNO [14]. The abstract OIDC protocol flow within Mobile Connect was presented in a past chapter [15].

4 - The ID GW of the operator sends a message to the app through the push messaging service (step 4.1). The user unlocks the app (step 4.2) in order to start the EAP AKA procedure (step 4.3). The ID GW of the MNO is responsible for the authentication of the end user. This is the entity that selects the authentication method. It also implements the OIDC protocol and generates the ID tokens [14]. In our case the ID GW selects as an authentication method the EAP AKA. As it is mentioned before, in order to transfer the EAP messages through the internet the HTTP authentication with EAP or the IPsec in tunnel mode could be used. The EAP AKA successful full authentication procedure is described in [10] and is presented below adjusted in the needs of our approach.





On the first EAP Request (EAP-Request/Identity) issued by the Authenticator, which located in the ID GW of the MNO, app ‘s response (EAP-Response/Identity) includes either the user ‘s International Mobile Subscriber Identity (IMSI) or a temporary identity (pseudonym), includes the Network Access Identifier (NAI) realm portion in environments where a realm is used.

After obtaining the subscriber identity, the EAP server, which is also located in the ID GW of the MNO, obtains an authentication vector (RAND, AUTN, RES, CK, IK), from an Authentication Center (AuC), for use in authenticating the subscriber. From the vector the EAP server generates the keying material. The EAP server sends an EAP-Request/AKA-Challenge message. The EAP-Request/AKA-Challenge message contains a RAND random number (AT_RAND), a network authentication token (AT_AUTN), and a message authentication code (AT_MAC). The app runs the AKA algorithm and verifies the AUTN. If this is successful the EAP server is legitimate and the app sends the EAP-Response/AKA-Challenge, which contains the result parameter (AT_RES) that allows the EAP server to authenticate the app and by extension the user and the AT_MAC attribute to protect the integrity of the EAP message.

The EAP server verifies that the RES and the MAC in the EAP-Response/AKA-Challenge packet are correct and sends the EAP-Success packet.

The above messages will be transferred as HTTP headers in base64 format, if the HTTP Authentication with EAP is the used method.

5 - After user ‘s authentication, the last step of the process is that the ID GW returns the response in the request that happens in step 2 depending on the OIDC grant-type used e.g. for Authorization Code grant-type, the Authorization Code is returned. The service provider gets the anonymized identifier and the authentication context [15].

The Mobile Connect checkout process completed successfully and the user now is able to proceed to buy the book.

Benefits/Advantages

The advantages that come with our approach have to do with the fact that the alternative methods supported by Mobile Connect either have security or complexity issues. The use of the EAP AKA within a SAA combines the usability of a SAA and the high level security of the EAP AKA protocol. It keeps the interaction of the user with the mobile device to the minimum, the only thing the user has to do is to have an enabled USIM card, install the application and configure the lock settings the first time that the app runs. When it is time to authenticate with mobile connect the user only has to unlock the app. Not only is the use of EAP AKA as an authentication method very convenient for the users, but also it is easy to be adopted by the MNOs. Furthermore, Mobile Connect authentication system employs private and secure mobile network operator channels instead of using the public internet. Our suggested methodology is secure even if the public internet is used, since the EAP AKA in combination with the HTTP

authentication with EAP or the IPsec in combination with EAP AKA, offers a secure way to transfer the EAP packets through the public internet and by extension authenticate the user. Also, in order to enhance the security, methods like EAP TTLS or PEAP could be used. More specifically, EAP TTLS and PEAP establish a secure TLS tunnel and inside this tunnel the EAP AKA authentication could take place. From the MNOs point of view, the benefits come with the fact that AKA is a protocol already known to them, as it has been used for many years in order to authenticate the users who try to access a 3G network. What is more, the MNOs have already the infrastructure and the knowledge in order to support the implementation of EAP AKA mechanism. Among other things, EAP AKA offers high level of security which has been tested all these years.

Possible Drawbacks

USIM Card

All the authentication methods which are supported by Mobile Connect have some drawbacks, some of them are less and others are more important. The solution we propose requires the existence of a USIM card, since the SIM card is the main component for the authentication of the user and the Authentication and Key Agreement mechanism is deployed in USIM cards. This could be a drawback for some users who might not have a USIM, as they will not be able to use this authentication method. At this time, it is difficult to find someone who does not have a USIM card, as USIM cards have been used for almost twenty years since the release of the 3rd Generation (3G) network. The USIM brought, among others, security improvements like the mutual authentication and longer encryption keys and an improved address book. Hence, it will be a good opportunity for those that do not have already have a USIM card to buy one. If someone does not want to change the old SIM card, there are alternative methods which could be used instead of EAP AKA. Thus, in case a user is not equipped with a USIM card, other EAP methods could be used for authentication like EAP TLS [20] and EAP TTLS [18].

IMSI Leakage

Another drawback that comes with our solution is the leak of the IMSI. IMSI stands for International Mobile Subscriber Identity, it is a sequence of numbers which is used for the user authentication in a mobile network, hence it is unique for every user. The IMSI is located in two places, in the SIM card and more specific in the read-only part and in the Home Location Register (HLR/Auc) of the MNO. It can have maximum fifteen digits.

- The first three are the Mobile Country Code (MCC).
- The next two or three digits represent the Mobile Network Code (MNC).
- The remain digits compose the Mobile Subscriber Identity Number (MSIN)
- The MNC along with the MSIN constitute the National Mobile Subscriber Identity (NMSI)

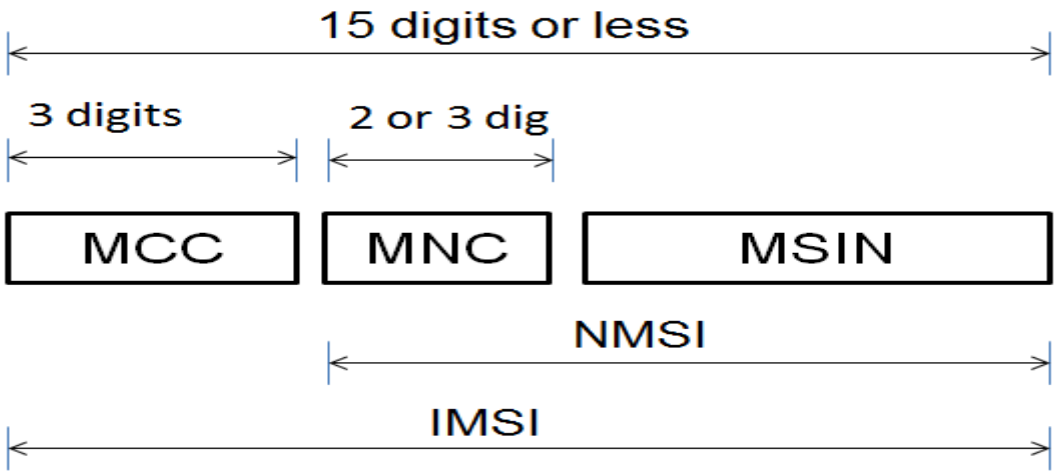


Figure 20: IMSI FORMAT

The EAP AKA includes a mechanism that protects the privacy of the subscriber ‘s identity. This mechanism delivers pseudonyms from the EAP server to the user as part of the EAP AKA exchange. A user that has not yet performed an EAP AKA exchange yet does not have a pseudonym available, which means that the privacy mechanism cannot be used and the permanent identity (IMSI) will have to be sent in clear text. Hence, this makes the IMSI visible to passive eavesdropping [10]. The same security issue has noticed with WiFi-calling (VoWIFI) service [16]. We can defeat this drawback by using a tunnel EAP method, like EAP TTLS [18], PEAP [19] etc. (e.g. EAP TTLS+EAP AKA). In case of a mobile phone loss or steal the user can report it in the operator and the Mobile Connect account will be blocked [17].

Conclusion

In the fast-paced days we live in, everybody wants to do everything as fast as possible; inability to do so (completing large forms with personal info, searching for forgotten credentials, etc.) may very well result in abandoning the process altogether. Combining Mobile Connect with EAP-AKA makes it much easier for the users to buy products or services online and avoiding situations like the ones mentioned before. Also, by combining the EAP AKA with the HTTP Authentication with EAP or the IPsec in tunnel mode, the EAP AKA implementation as an authentication mechanism within Mobile Connect, will be achieved easier without significant changes in the existing infrastructure of the MNO. That benefits businesses that support their revenue on sales made online, and MNOs already have the infrastructure ready to go, which means that the cost for them will be minimal. Moreover, it offers a very good level of security and the probability of a user getting compromised is incredibly low. The drawbacks we mentioned above are negligible compared to the advantages this solution offers.

References

[1] GSMA Mobile Connect - https://www.gsma.com/identity/mobile-connect
[2] GSMA - CPAS04, Authenticator Options, Version 1.0, 11 November 2015
[3] 3GPP TR 22.934: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Feasibility study on 3GPP system to Wireless Local Area Network (WLAN) interworking".
[4] 3GPP TS 33.234 V6.9.0 (2007-03)

[5] GSMA Position Paper for Web Cryptography Next Steps Workshop, Hardware Tokens: SIMApplets for use in Mobile Connect

[6] Digital cellular telecommunications system; Unstructured Supplementary Service Data (USSD) - Stage 2

[7] https://developer.mobileconnect.io/mobile-connect-api

[8] MePIN https://www.mepin.com/mobile-connect/

[9] Mobile ID https://www.gemalto.com/mobile/id-security/mobile-id
[10] J. Arkko, Ericsson, H. Haverinen, Nokia. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). RFC 4187. 2006
[11] API Exchange - https://www.gsma.com/identity/api-exchange
[12] OpenID Connect - http://openid.net/specs/openid-connect-core-1_0.html#CodeFlowAuth
[13] D. Hardt, Ed. Microsoft. The OAuth 2.0 Authorization Framework. RFC 6749, 2012
[14] GSMA Mobile Connect MNO Implementation Requirements, Version 0.1
[15] GSMA CPAS 5 OpenID Connect – Mobile Connect Profile, Version 1.1
[16] Piers O' Hanlon, Ravishankar Borgaonkar, Wi-Fi Based IMSI Catcher, 2016.
[17] Mobile Connect - https://mobileconnect.io/
[18] P. Funk, Unaffiliated, S. Blake-Wilson, Safenet. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0). RFC 5281, 2008.
[19] Ashwin Parker, Dan Simon, Microsoft, Glen Zorn, Cisco, S. Josefsson, Extundo. Protected EAP Protocol
[20] D. Simon, B. Aboba, R. Hurst, Microsoft Corp. The EAP-TLS Authentication Protocol, RFC 5216, 2008
[21] Bjorn Hjelm Verison, John Bradley Yubico, The interface of MODRNA (Mobile Profile of OpenID Connect) and GSMA Mobile Connect, 2017 - http://openid.net/wg/mobile/
[22] J. Arkko, V. Torvinen, Ericsson, A. Niemi, Nokia, HTTP Authentication with EAP, 2001
[23] https://baymard.com/lists/cart-abandonment-rate

[24] https://www.barilliance.com/cart-abandonment-rate-statistics/

[25] https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[26] IMSI - https://en.wikipedia.org/wiki/International_mobile_subscriber_identity

[27] B. Aboba Microsoft, L. Blunk Merit Network Inc., J. Vollbrecht Vollbrecht ConsultingLLC, J. Carlson Sun, H. Levkowetz, Ed. IpUnplugged. Extensible Authentication Protocol (EAP). June 2004

[28] JSON Web Token - https://jwt.io/

[29] M. Jones Microsoft, J. Bradley Ping Identity, N. Sakimura NRI, JSON Web Token (JWT). May 2015

[30] OpenID Foundation - http://openid.net/foundation/

[31] IPsec - https://en.wikipedia.org/wiki/IPsec

[32] https://realtimecommunication.wordpress.com/2016/04/06/epdg-and-ipsec/

[33] P. Eronen Independent, H. Tschofenig Nokia Siemens Networks, Y. Sheffer Independent. An Extension for EAP-Only Authentication in IKEv2. RFC 5996, September 2010.

[34] EAP packet format – http://etutorials.org/Networking/Wireless+lan+security/Chapter+7.+EAP+Authentication+Protocols+for+WLANs/EAP/

[35] https://www.buzzfeed.com/josephbernstein/survey-says-people-have-way-too-many-passwords-to-remember?utm_term=.gsWoxvGZ9#.xg8nXb4yY