User-Centric Identity Management Using Trusted Identity Module - Binding Mobile Phone Secure Elements to the OpenID Connect Protocol

Johann Vincent (johann.vincent@orange.com)*
Sahin Kale (sahin.kale@orange.com)*
Vincent Frey (vincent.frey@orange.com) *

Abstract: Following the massive adoption of new platforms such as Android or iOS, smartphone security research has become a very popular topic. Nonetheless, the issue of authenticating the user with its applications, in order to provide him with personalised services, is still open. In this paper, we present the Trusted Identity Module (TIM), a local smartphone module that enable the user to log into application using the newly proposed OpenId Connect protocol. This TIM uses an active cardlet installed on a secure element such as the SIM/UICC of the mobile phone to store long-lived tokens and perform cryptographic operations. We show that our TIM solution improves usability, security and privacy protection for the user with few impacts on the still emerging OpenId Connect protocol.

Keywords: Digital identity, Trust, Privacy, OpenId Connect, Single Sign-on

1 Introduction

With the advent of electronic networks and particularly the Internet, new services have appeared. Nowadays, it is possible to shop online, consult bank accounts, share information on social networks or to host documents and programs on cloud platforms. With such services, the creation of a trust relationship, between these services and the end user have become a challenging issue. In [LS09], the authors call this relationship a trusted path and give the following definition : "this path can take many forms, but the characteristics we want are that the user can be sure they are talking to the correct server, can see what that server intends them to see, that the server can be sure that the user sees what it wants shown, and that the actions taken by the user are faithfully reflected to the server". One of the way to establish a trust relationship between two entities is to use predefined policies. This policy-based trust [AG07] assumes that trust is established when an entity has received a sufficient amount of credentials regarding another entity and applies a specific policy to grant that entity certain access rights. One of the historical, and still vastly deployed, credential is the couple login/password. However, with the multiplication of services and the multiplication of the means of access, this situation has become a burden for the end user. For the last 10 years, many solutions, so called Identity Management Systems (IdMS)

∗. Orange Labs, Cesson-Sévigné

have been proposed to change the password paradigm.

In [BHvOS12] and [Vin13], the authors try to provide an overview of these solutions as well as a mean to evaluate them in regards of security and privacy. One of the conclusion they draw is that no system is perfect. However, some models or combination of models seem to offer a good compromise. The user centric model using an intelligent client, such as SAML2 Enhanced Client Profile [HCH+05] or ID-WSF advanced client [All07a], provides good security as well as good privacy protection.

Unfortunately, these models have not been deployed on a vast scale. First because they have not been adopted by big service providers, so called Over-The-Top (OTT) and secondly because they raise another issue that is trust in the client and the system that runs them. This latter issue has been studied on PC platforms by the Trusting Computing Group resulting in the specification of the Trusted Platform Module (TPM) [TCG03]. However, since 2011, the smartphones sales have outnumbered the pc sales and share now nearly 78% of the market [gar13]. More and more people use them to access the services presented above and they have become the swiss army knife of user's digital life. Meanwhile, the mobile OSes remains untrustworthy and some efforts have been made to implement trusted architecture on mobile platforms as well. In [VOZ+12], the authors give an overview of trustworthy execution on mobile devices. It details the security properties of isolated execution, secure storage, remote attestation, secure providing and trusted path for a device. It then presents some of the available hardware primitives that provides the security features presented above and the API that specify both how to write secure modules and how an untrusted application can interact with such module.

In this paper, we propose to consider these later findings with a solution that implements the user-centric identity management model with a local trusted client on smartphones that we call Trusted Identity Module (TIM). To do so, we propose a modification of the newly proposed OpenId Connect protocol [S+11] that is becoming a large standard among OTTs and we enforce security in the client by coupling it to a local Secure Element (SE). The paper is organised as follows : the related works are presented in the first section, then in the second section, the TIM proposal is presented. We show in the third section that it provides better security as well as better privacy protection and usability for the end-user. Finally, the paper is concluded and some perspectives and future works are proposed.

2 Related works

As said in the introduction of the paper, among the many identity management systems, the ones that implement the user-centric model with a local client seem to offer the best compromise between security, privacy and usability. They describes the exchange of data between four majors actors which are the subject, the client, the service provider(SP) and the Identity Provider (IdP). In this section, a review of such solutions for mobile phone is given. We first detail the use of a local client application for well known IdMS solutions and then present the efforts made to use hardware backed trusted module on smartphone.

2.1 Smartphone local identity client for IdMS

The two OTTs Google and Facebook propose a way to authenticate the user on mobile applications by using their identity services. They both have developed mobile APIs that allows an application to run the OAuth 2.0 protocol [HLRH11] between the application (which becomes an Oauth client) and their authorisation server. In the case of *Google+ sign-in* on Android, the API relies on the Google Play service that is running on the phone. On other platforms (and for the Facebook case) the client application can use the local Google+ or Facebook application as a proxy instead of the Google Play Service. These applications (or service) maintain a session for the user and make calls to the online services to get updates (on access tokens for example). They however work only when the device is connected and the online services are available. The BrowserId protocol [Fou13] has been proposed by the Mozilla Foundation in 2011 and allows a user to register and/or log into websites by providing, via his user-agent, a signed assertion of email ownership. The protocol can be separated into three steps : the user certificate provisioning, the assertion generation and the assertion verification. The main difference between this approach and the other that we have presented is that the identity provider (IdP), here the email provider, is not aware of the user's transactions. In fact, as soon as the user agent have received a valid certificate, it can use it to generate identity assertion without contacting the IdP. The only contact that can occur is when the service provider (SP) contacts the IdP to get its public key in order to verify an assertion. But even when doing so, no information about the user is transmitted. The most important criticism made about BrowserId is that it is based on a compatible web browser running on an untrusted device [Cla]. Finally, an OpenId Connect working group has started to work recently on the concept of Native Authorisation agent (previously known as AZA [MJZ13]) which is similar to the Google services only for the specific OpenId Connect protocol. Their Authorisation agent act as a proxy for OpenId connect request to online Authorisation Servers to create a local SSO for native applications. The specification mention that the verification of tokens should be possible without contacting the online Authorisation Server in order to protect the user's privacy. However, no specific mechanism is given to achieve such a goal. These solutions raise two problems that are the privacy risks of having to contact the online IdP and the problem of trust in the local application or service. To address that later issue, some efforts have been made on using local Secure Element (SE) or Trusted Execution Environment (TEE) as demonstrated in the next paragraph.

2.2 Secure Element and Trusted Execution in main IdMS

The main efforts made on using a trusted hardware module on mobile phone to enhance existing IdMS have been pursued by the Liberty Alliance consortium and the 3GPP organisation. The 3GPP has defined the Generic Authentication Architecture (GAA) independent of other identity management systems. It offers a mechanism to provide a shared secret and certificates to two communicating entities for mobile application, based on GSM and UMTS authentication and key agreement protocols. The two technical reports [3GP09] and [3GP11] describe a way to use GAA in respectively SAML2 and OpenId. These reports define two solutions for inter working : the collocation of the IdP and the Bootstrapping Server Function (BSF) (for SAML2 only) and the collocation of the IdP and the Network Application Function (NAF). The Liberty Alliance has proposed in [All07a] an advanced client which has a component called Trusted Module (TM) that

would work as an extension of the online IdP. That TM supports two modes : a connected mode where the TM is here to facilitate identity operations and a disconnected one where it is autonomous. In [All07b], the authors propose an implementation for this TM in a SIM/UICC card. The TM is implemented as a JavaCard cardlet which is exposed through a SmartCard Web Server and is provisioned using the GAA mechanism mentioned above. In [LSS12], the authors present Smart OpenId which introduce a local IdP similar to the Liberty Alliance approach with the Trusted Module. This local IdP (or local OP) is in charge of authenticating the end user and issuing the OpenId assertions. The local IdP they propose is implemented by a combination of a user level application which provide HTTP interface and an applet on the UICC. The author also introduce an OpenId support function (OPSF) which is operated by the MNO and is reachable by the SP. This OPSF needs to share a secret with the local IdP situated on the UICC, that can be done by using the operator's capabilities described earlier.

The addition of smart cards and especially SIM/UICC is used both to provide a trusted environment and to enhance the authentication of the user by using the operator's infrastructure. One of the thing that is to be noted is that most of the efforts were made by the Liberty Alliance consortium. This is the reason why they're is, to the best of our knowledge, no effort on using trusted environment in newer IdMS such as OpenId connect or OAuth 2.0. The approaches proposed by Liberty Alliance also suppose a strong connection between MNOs and IdPs or that the MNO plays the role of an IdP. In the current context, it is unlikely that the main IdPs will find a need of such a strong association with an MNO unless it can increase the trust put in their own identities. Another problem that is to be noted is that no solution provides a secure UI that is necessary for a complete trusted path from the user to the services.

3 Trusted Identity Module proposal

In this section the Trusted Identity Module (TIM) is presented. First, we detail the awaited usability, security and privacy protection requirements for a Trusted Identity Module. Then the TIM architecture is presented and finally we show how it can provide the requirements.

3.1 Requirements

Single Sign-on. Single Sign-On (SSO) allows a user to log to multiple applications and services without having to re-enter his credentials. This is both a usability and security requirement for the TIM as it lightens the burden of memorising multiple passwords and limits the risks of phishing attacks.

Usage continuity. The usage continuity requirement is a usability requirement that makes it possible for a user to use his TIM while changing his network access. For example, when switching from 3G/4G network to Wi-Fi or even in a disconnected scenario.

Secure credential storage. The user's credentials should be stored in a tamper-resistant element and no identity information nor session token should be store on the

device. The access to this secure storage should be strictly controlled and only granted to the TIM and under the control of the end-user.

Isolated execution. Isolated execution allows the TIM to run in complete isolation from other code. It provides secrecy and integrity to the TIM code. Today's OS provides process based isolation where applications runs in sandboxes. However, these solutions are not sufficient when the OS is compromised which can occur easily in the case of rooted devices for example.

Trusted Path. As presented in [VOZ+12], a trusted path is a secured path between a software module, here the TIM, and the peripherals. It is important to ensure that the user is entering his credential in a TIM controlled screen and that the access to the secure storage is only possible from the TIM.

Privacy : Unlinkability. In most current IdMS, the IdP is an active participant which makes it a central point that can track the user's activity online. In fact, by construction, it can track every use of the user's identity and as such it can easily keep a record of the visited service providers. The unlinkability from the IdP point of view is the requirement that such tracking is not possible by the IdP. However, one might also want to prevent service providers to track their activity. That means that a service provider is not able to tell from the authentication whether two sessions are related to the same user. In this paper we call partial unlinkability the unlinkability from the IdP point of view.

3.2 The TIM

The TIM is a mobile service that serves as an identity proxy on a mobile device for installed applications. The TIM is based on the OpenId Connect [S+11] specifications that provides an identity layer on top of the OAuth 2.0 protocol [HLRH11]. OpenId Connect allows web-based, native and JavaScript client to request and receive information about authenticated sessions and end-users. The actors present in our architecture are the same as in OAuth 2.0 with the addition of the TIM :

- The resource owner (the end-user),
- The mobile application that want to access some identity information on the end user,
- The Trusted Identity Module (TIM),
- The online Authorisation Server (AS [1]),
- The Resource Server.

In order to be able to issue requests to the OpenId provider's AS, the TIM must have registered itself as a client and authorised party with the AS. The registration process of the TIM is not in the scope of this article, it can occur either by out-of-band mean or by following the OpenID Connect Dynamic Client Registration 1.0 [SBJ]. We assume in the rest of the article that the TIM `client_id` has been registered, that its public parameters (public keys) are known to the AS and that the TIM knows about the AS public parameters. We also assume that the private parameters for each of the parties are kept secret

1. Not to be mistaken with the Application Server that can be used in IP Multimedia Subsystem (IMS)

and, in the case of the TIM's ones, inside a secure element such as the SIM/UICC.

In our solution, the mobile application (*app*) uses the TIM API to create authorisation request to the Authorisation Server (thus reducing the burden of having to implement a full OpenId connect client). We have chosen to implement the authorisation code flow of the OpenId connect protocol as it allows the mutual authentication of the actors. It is triggered when a user chooses log-into the application, which call the TIM services with the authorisation request parameters [2] for the application. Upon the reception of this request, the TIM checks if it has a local identity that can match the request, if not, it creates its own authorisation request to the online Authorisation Server. This new request contains the same scope as the application one with the addition of the `tim` scope. It also contains a new request parameter that contains an additional `tim` claim. This request parameter is signed by the TIM and encrypted using the AS public key. When the OpenId Authorisation Server receives the request, it validates it and if needed authenticates the user and obtains his consent for a TIM usage. The user agent for this is managed by the TIM which means that session cookies are under the control of the TIM and not available to other applications and other distant untrusted sites. The Authorisation Server replies with an authorisation code in compliance with the OpenId Connect specification. The TIM then requests an `access_token` and an `id_token` with the authorisation code. The online AS replies with three tokens :

- A temporary `access_token`,
- A long-lived `refresh_token`,
- An `id_token` in which the audience claim contains both the TIM and the Application `client_id` and the authorised party claim contains the TIM `client_id`.

In order to be able to issue its own token for the application, the TIM computes a new key pair inside the secure element that will be used respectively to sign and verify TIM's access tokens. Next, the TIM needs to send the public key of the pair to the Authorisation Server (similar to what is done in Mozilla Browser Id [Fou13]). To do so, it uses its `refresh_token` with an additional `tim_app_key` parameter. When the AS receives the public key, it delivers another `id_token` that contains the `tim_app_key` and that will be used for verification. This `id_token` acts as a certificate for the key associated with this application and TIM. The goal is to allow the validation of `access_token` by verifying the signature without having to contact the AS. To finish the login process, the TIM sends back an authorisation code to the application that uses it to request the access and id tokens. The TIM computes a new access token and signs it with its key and sends both this new `access_token` and the `id_token` to the application. The TIM is coupled with an applet inside the Secure Element. All the cryptographic operations are done inside that applet and long-lived tokens (id and refresh) as well as keys are stored inside this applet. The whole protocol is detailed on figure 1.

4 TIM validation

While there are some attempts at formally verifying the Oauth 2.0 protocol [BBM12], this kind of study has not yet been conducted on the OpenId Connect protocol and was not in the scope of our current work. In this section, we present how our TIM solution

2. response_type, client_id, scope and redirect_uri

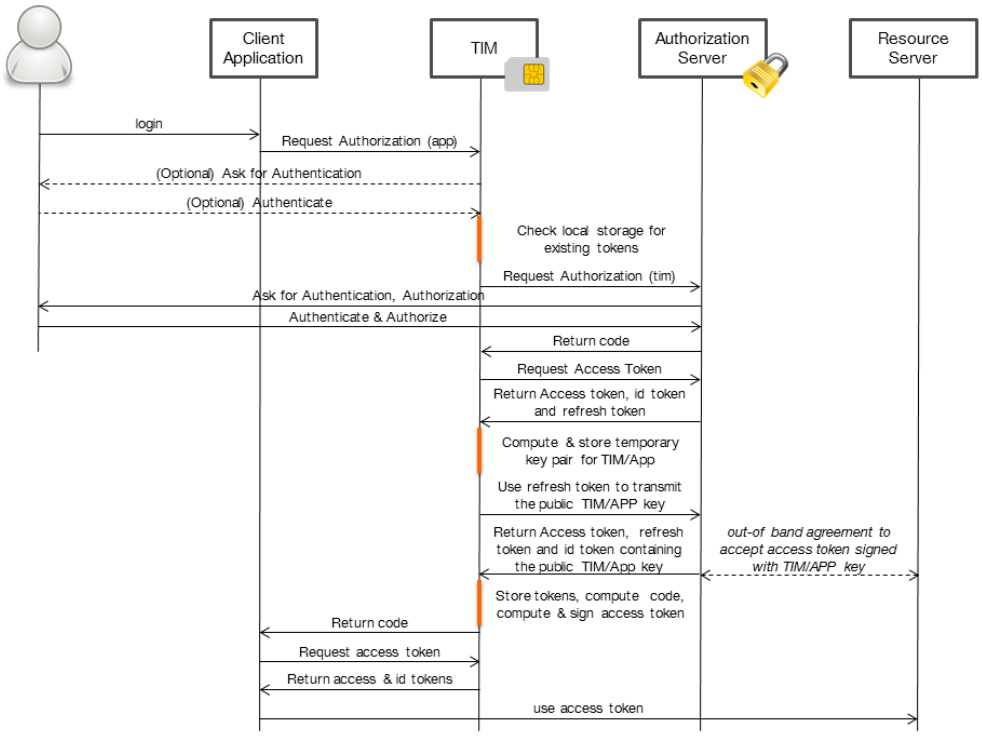

Figure 1: TIM sequence diagram

addresses the usability, security and privacy protection requirements we have presented.

Single Sign-On. First, as the TIM act as a proxy for OpenId Connect requests, if two applications request identity information to a given OpenId provider, the user will only have to sign-in once. This is achieved for two reasons, if the TIM has valid local `id_token` it can issue valid tokens for the application. If on the other hand the TIM has no valid token, it will have to redirect the user to the AS. The TIM acts as a web browser in this case and as such it handles a session cookies for this provider which is stored securely in the SIM/UICC. By doing so, it is not necessary for the user to authenticate himself with the AS during the cookie's lifetime.

Usage Continuity. It will also only have to perform the described protocol once during the lifetime of the `refresh_token` for each application thus reducing the need for a data connectivity. This means that after the first retrieval of tokens the TIM can work in a scenario where it is disconnected from the AS and deliver valid tokens to applications.

Secure Credential Storage and Isolated Execution. The use of a cardlet inside a Secure Element grants the secure storage properties for long lived information : `id_tokens`,

`refresh_token`, `tim_app_keys` and TIM specific keys. This SE also provides a trusted execution environment for all the cryptographic operations that have to be done by the TIM, which are the encryption and signature of JSON Web tokens.

Trusted Path. As mentioned above,in order to send these tokens, the TIM must also have a high level component that can use the data connectivity and create HTTPS requests to the online AS. This component is also in charge of the interaction with the end-user. In the presented TIM, these interactions are not secured, it means that there is no trusted path between the TIM and the peripherals (here the touch screen) and a malware could mimic the TIM behaviour to trick the end user into entering his credential on a malicious site or application. In the next paragraph, we propose a possible improvement of the TIM solution to address that later issue.

Partial Unlinkability. Finally, the fact that the TIM generates its own keys to sign access token prevents the online AS to track the user's activity as any resource server can check the validity of the access token by validating the TIM signature and does not need to contact the AS each time. However, this only provides the partial unlinkability property , as the resource server can track the user's activity by tracking the access token signatures that are linked to the TIM and thus to the user's device. The summary of the evaluation is shown in Table 1

Requirements	Google+ sign-in	Mozilla BrowserId	GAA OpenId & SAML2	**TIM**	**TIM (with TEE)**
Single Sign-On	✔	✔	✔	✔	✔
Usage continuity	✖	✔	✖	✔	✔
Secure storage	✖	✖	✔	✔	✔
Isolated execution	✖	✖	✔	✔	✔
Trusted path	✖	✖	✖	✖	✔
Partial unlinkability	✖	✔	✖	✔	✔
Unlinkability	✖	✖	✖	✖	✖

Table 1: Comparison of the solution in regards of the requirements

4.1 Possible improvements

With the described TIM, the main requirement that is not fulfilled is the trusted path between peripherals (in particular the touch screen) and the end-user. In the current ecosystem, it is hard to provide that requirement but in this paragraph we try to give some insights regarding that issue. The Global Platform consortium provides a set of

standards for a trusted execution environment (TEE) [Pla11] which describe both an API to interact with secure module executing inside the TEE and the way to write such module. These standards combined with hardware extensions such as ARM TrustZone [AF04] can provide a foundation for a trusted path between the TIM and the peripherals. While this perspective has been discussed in previous works [VOZ+12], to our knowledge the currents TEE implementations (for example the Trustonic one) do not support it yet and it is still an active research field and technical challenge.

5 Conclusion and perspectives

The usage of the OAuth 2.0 protocol and its authentication counterpart OpenId Connect is spreading on smartphone devices. In this paper, we have presented the Trusted Identity Module (TIM) which enable the user to securely log into mobile applications using the OpenId Connect protocol while providing multiple benefits among which : single sign-on, usage continuity, secure storage, isolated execution and partial unlinkability. We have shown that the TIM only requires few modifications of the OpenId Connect protocol which is the addition of a `tim_app_key` claim. We have also shown that this solution provides more benefits for the privacy protection of the end-user than the existing ones.

Future works will consist in the formal validation of our protocol following the formal validation of the OpenId Connect protocol. This validation is to be followed by the implementation of a complete TIM prototype on an Android device using the SIM/UICC as a secure element. This prototype will also help us validates the solution in term of performances and it will allow us to test the usability impact of this solution. We intend to propose a standard API for applications so that they can use the TIM to perform login operation with external identity providers. Finally, the study of a TIM implemented on a TEE is also one of the perspective of this work as well as solutions to provide full unlinkability, such as the use of group invariable partially blind signature [CRJ13].

Références

[3GP09] 3GPP. Interworking of liberty alliance identity federation framework (id-ff), identity web services framework (id-wsf) and generic authentication architecture (gaa). Technical report, ETSI, 2009.

[3GP11] 3GPP. 3gpp : Identity management and 3gpp security interworking ; identity management and generic authentication architecture (gaa) interworking. Technical report, ETSI, 2011.

[AF04] Tiago Alves and Don Felton. Trustzone : Integrated hardware and software security. *ARM white paper*, 3(4), 2004.

[AG07] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semantics : Science, Services and Agents on the World Wide Web*, 5(2) :58–71, 2007.

[All07a] Liberty Alliance. Id-wsf advanced client 1.0 specifications. Technical report, Technical report, 2007.

[All07b] Liberty Alliance. Id-wsf advanced client implementation and deployment guidelines for sim/uicc card environment. Technical report, Technical report, 2007.

[BBM12] Chetan Bansal, Karthikeyan Bhargavan, and Sergio Maffeis. Discovering concrete attacks on website authorization by formal analysis. In *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*, pages 247–262. IEEE, 2012.

[BHvOS12] Joseph Bonneau, Cormac Herley, Paul C van Oorschot, and Frank Stajano. The quest to replace passwords : A framework for comparative evaluation of web authentication schemes. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 553–567. IEEE, 2012.

[Cla] Roger Clarke. Reactions to mozilla's browserid proposal.

[CRJ13] Sebastien Canard, Lescuyer Roch, and Traoré Jacques. More privacy for identity federation. 2013.

[Fou13] Mozilla Foundation. Browserid specification, 2013.

[gar13] Gartner says worldwide pc, tablet and mobile phone combined shipments to reach 2.4 billion units in 2013, 2013.

[HCH+05] John Hughes, Scott Cantor, Jeff Hodges, Frederick Hirsch, Prateek Mishra, Rob Philpott, and Eve Maler. Profiles for the oasis security assertion markup language (saml) v2. 0. *OASIS Standard*, 2005.

[HLRH11] E Hammer-Lahav, D Recordon, and D Hardt. The oauth 2.0 authorization protocol. *Network Working Group Internet-Draft*, 2011.

[LS09] Ben Laurie and Abe Singer. Choose the red pill and the blue pill : a position paper. In *Proceedings of the 2008 workshop on New security paradigms*, pages 127–133. ACM, 2009.

[LSS12] Andreas Leicher, Andreas U Schmidt, and Yogendra Shah. Smart openid : A smart card based openid protocol. In *Information Security and Privacy Research*, pages 75–86. Springer, 2012.

[MJZ13] P. Madsen, A. Jain, and A. Zmolek. Openid connect native authorization agent token provisioning profile 1.0, 2013.

[Pla11] Global Platform. The trusted execution environment : Delivering enhanced security at a lower cost to the mobile market, 2011.

[S+11] N Sakimura et al. Openid connect standard 1.0, 2011.

[SBJ] N Sakimura, J Bradley, and M Jones. Openid connect dynamic client registration 1.0-draft 23, december 2013.

[TCG03] PC TCG. Specific implementation specification version 1.1, 2003.

[Vin13] Johann Vincent. *Identité numérique en contexte télécom*. PhD thesis, Université de Caen Basse-Normandie, 2013.

[VOZ+12] Amit Vasudevan, Emmanuel Owusu, Zongwei Zhou, James Newsome, and Jonathan M McCune. Trustworthy execution on mobile devices : What security properties can my mobile platform give me ? In *Trust and Trustworthy Computing*, pages 159–178. Springer, 2012.