D7.6 Final implementation of the privacy enhanced digital currency prototype

Document Identification			
Status	Final	**Due Date**	30/09/2020
Version	1.0	**Submission Date**	30/09/2020

Related WP	WP7	**Document Reference**	D7.6
Related Deliverable(s)	D7.1, D7.2, D7.3, D7.5, D7.6, D7.7, D7.8, D7.9, D7.10, D7.11, D7.12	**Dissemination Level (*)**	PU
Lead Participant	ATOS	**Lead Author**	Miguel Angel Mateo (ATOS)
Contributors	ATOS	**Reviewers**	Clément Geentilucci (FUAS)
			Alain Paschoud (KUD)

Keywords:
Final Version Demonstrator Report, technical specification, use case, digital currency, crypto API

This document is issued within the frame and for the purpose of the FENTEC project. This project has received funding from the European Union's Horizon2020 under Grant Agreement No. 780108. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

List of Contributors	
Name	Partner
Raquel Cortes Carreras	ATOS
Miguel Angel Mateo	ATOS

Document History			
Version	Date	Change editors	Changes
0.1	21/05/2020	ATOS	First Version
0.2	23/09/2020	ATOS	Ready for internal review
0.3	28/09/2020	ATOS	Internal review comments addressed
1.0	28/09/2020	ATOS	Final version

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Miguel Angel Mateo (ATOS)	30/09/2020
Technical Manager	Michel Abdalla (ENS)	30/09/2020
Quality Manager	Diego Esteban (ATOS)	30/09/2020
Project Coordinator	Francisco Gala (ATOS)	30/09/2020

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
ABE	Acronym description
API	Application Programming Interface
CP-ABE	Ciphertext Policy Attribute-Based Encryption
DB	Data Base
FE	Functional Encryption
KP-ABE	Key Policy Attribute-Based Encryption
REST	REpresentational State Transfer

Executive Summary

This deliverable describes the last version of the ATOS Privacy-Preserving digital currency use case prototype. This version has been developed as demonstration of viability of the pilot from different points of view; cryptographic, performance and functionality. We conclude that the Functional Encryption schemes developed in the project provides a tool easy to integrate and with a performance adequate for this pilot. Performance results obtained will be described in deliverable D7.12.

1 Introduction

This deliverable aims to present the last version of the source code of the privacy enhanced digital currency prototype. This development follows the Requirement analysis presented in D3.1[1] and the specification done in D7.1[3]. Regarding cryptographic protocols, they were fully explained in D7.5, therefore, we focus this deliverable on the release of software.

This deliverable is produced concurrently with the development of the prototypes for the other two use-cases and leads into the reports on performance and security:

D7.1 Preliminary specification of FENTEC prototypes (M9)

D7.2 Final specification of FENTEC prototypes (M25)

D7.3 First version of the truly anonymous data collection prototype (M22)

D7.4 Final Version of the Truly Anonymous Data Collection Prototype (M33)

D7.5 First version of the privacy enhanced digital currency prototype (M22)

D7.7 First version of the IoT key distribution prototype (M22)

D7.8 Final version of the IoT key distribution prototype (M33)

D7.9 First test report of the FENTEC prototypes (M23)

D7.10 Final test report of the FENTEC prototypes (M34)

D7.11 Performance report for FENTEC prototypes after first cycle (M24)

D7.12 Final performance report for FENTEC prototypes after second cycle (M36)

2 Privacy-Enhanced Digital Currency Demonstrator

This deliverable takes as base the work presented in D7.5, following the crypto protocol described in that deliverable. In contrast with the source code delivered together with D7.5, where it was a single application, the current version of the pilot has been developed as a collection of WEB services to implement each of the entities involved in the pilot operation: Customer, Merchant, Trusted Authority and Exchange service. Each of the entities provide a REST API with the functionality needed to demonstrate the basic operation of the pilot and how crypto protocols work.

2.1 Changes from last version

2.1.1 Integration with MONGO DB

During the performance test of the last version we experimented issues due to the RAM memory needed to host all concerned documents during a basic purchase operation. We must point that for each purchase, there are several copies of the invoice hosted by each of the entities participating in that operation, mainly customer and merchant, and in different formats; JSON, byte arrays, cyphered and not cyphered, etcetera. This invoice includes big amount of information due to eCoins and big data fields. In order to tackle this issue, we have integrated all the entities of the pilot with a MONGO DB that works as permanent repository for invoices of finished purchases and also as volatile repository for operations in progress.

2.1.2 Distributed Architecture

The four main entities of the demonstrator have been developed as WEB services using Maven to manage library dependencies, taking the previous development as starting point. The current version of these services has been developed to run in TOMCAT server.

Each of these services have a REST API with the operation required to perform each step of a purchase from the point of view of a user, hiding technical complexities.

2.1.3 Distributed Architecture

The pilot provides wrappers for the CP-ABE and KP-ABE schemes developed in WP6. These wrappers have been updated to the last version of this schemes which extend their functionality and provides new API.

2.1.4 Audit functionality

The Audit functionality has been developed and integrated into the Exchange server. It provides the basic operations to demonstrate how CP-ABE scheme can be used in order to keep the privacy of customer data.

This functionality has been initially integrated into the exchange as it is enough to show how it works, although, it should be developed as independent service in the future. The whole pilot continues to be developed in FENTEC in order to provide the most valuable demo at the end of the project.

2.2 Source Code

The source code is organized in five maven projects in order to lighten each of the phases of the project and manage the different libraries and packages used. These projects correspond to the four entities: Customer, Merchant, Exchange and Trusted Authority, plus a library with all common operations such as cryptographic protocols, data base connection, etcetera.

The source code is available at: https://scm.atosresearch.eu/fentec_digcur_uc_pub

2.3 APIs

Each of the four entities provides a REST APIs with different sets of operations for the interaction with users and the internal operation of the platform. Although most operations are named in this document, only those supposed to be invoked from the user side have an example of use. The rest of the operations are dedicated to the internal work between platform components. In these cases, bodies of requests contain cyphered data, signatures and cryptographic data structures, which are sent as serialized java objects.

2.3.1 Customer_service

This maven project implements the customer service and all the required encrypting operations related to this entity. It offers a REST API to perform the following operation.

- createUser

Name	Resource	Method	Description
createUser	/user	POST	Create customer and all related data structures required.
Example	POST http://95.211.XXX.XXX:8080/customer_service/customer/fentec/user		
Request	{ "nameList" :"Michael",		

	"ect":["Transport"] }	

Answer	200 OK	{
		"wallet": {
		"eCoinTypeAL": [],
		"funds": "500",
		"id": "71h/wRSezRquQsCwSUAkpg=="
		},
		"name": "Michael",
		}

Figure 1. Create user method

- withdrawal

Name	Resource	Method	Description
withdrawal	/withdrawal	POST	Exchange funds into eCoins
Example	POST http://95.211.XXX.XXX:8080/customer_service/customer/fentec/ withdrawal		

Request	{
	"name" :" Michael ",
	"ectName":"Transport",
	"ammount":"40"
	}		

Answer	200 OK	{
		"wallet": {
		"eCoinTypeAL": [{
		"ecoinAL": [
		{
		"id":
		"8l38O5u2tusppNBXhSRSd4V/dzUnzwbwjtXy3nN669s=",
		"value": "20"
		},
		{
		"id":
		"vIVyGbGa03H2RIaxQOuyzqCSZGlCRE95jxH3DcFFZBM=",
		"value": "20"
		}
		],
		"spent": "",
		"name": "Transport"
		}],
		"funds": "460",
		"id": "71h/wRSezRquQsCwSUAkpg=="	

		}, "name": " Michael ", }

Figure 2. Withdrawal method

- getCustomerDetails

Name	Resource	Method	Description
getCustomer Details	/user/[user name]	GET	Create customer and all stuff related to it operation such as wallet instance
Example	GET http://95.211.XXX.XXX:8080/customer_service/customer/fentec/user/Michael		
Request			
Answer	200 OK	{ "wallet": { "eCoinTypeAL": [{ "ecoinAL": [{ "id": "8l38O5u2tusppNBXhSRSd4V/dzUnzwbwjtXy3nN669s=", "value": "20" }, { "id": "vIVyGbGa03H2RIaxQOuyzqCSZGlCRE95jxH3DcFFZBM=", "value": "20" }], "spent": "", "name": "Transport" }], "funds": "460", "id": "71h/wRSezRquQsCwSUAkpg==" }, "name": "Michael", }	

Figure 3. Get customer details method

- purchase:

Name	Resource	Method	Description
purchase	/purchase	POST	this method is used to initiate a purchase operation by providing to the customer service a list of goods the user desires to buy at a merchant site
Example	GET http://95.211.XXX.XXX:8080/customer_service/customer/fentec/purchase		

Request	{
	"name":"Michael",
	"merchantName":"EMT",
	"products":[
	{
	"name":"Train ticket",
	"amount":"2"
	}
	]
	}		

Answer	200 OK	{
		"contractNumber":"1359596144",
		"issuer":"EMT",
		"cost":20,
		"taxes":4.2, "items":
		[
		{
		"name":"Train ticket",
		"amount":2
		}
		]
		}	

Figure 4. Purchase method

- payment:

Name	Resource	Method	Description
payment	/payment	POST	To pay a purchase
Example	GET http://95.211.XXX.XXX:8080/customer_service/customer/fentec/payemnt		

Request	{
	"name":"Michael",
	"merchantName":"EMT",
	"ectName":"Transport",
	"contractNumber":"1359596144"
	}	

Answer	200 OK	

Figure 5.: Payment method

2.3.2 Exchange server

This maven project implements the operation of the Exchange entity, providing a REST API dedicated to the internal platform interwork, with the following operations:

2.3.2.1 eCoin types management

- createEcointType: add a new eCoinType the Exchange entity can manage.
- getEcoinTypes: retrieve all eCoinTypes managed by an Exchange entity.
- getEcoinType: retrieve details regarding a specific eCoin Type.

2.3.2.2 eCoin management

- wireTransfer: this method transfers an amount of cash to an eWallet. If the wallet id does not exist at the exchange, it is created.
- getBlindSignature: this method is used to obtain a blinded signature for the ID of a new eCoin.
- verifyEcoin: this method is invoked to verify the validity of an eCoin, it verifies the validation token ID and its signature, and performs verification to avoid double spending fraud.

2.3.2.3 eInvoices management

- uploadeInvoice: this method is used to upload a cyphered eInvoice to the DB.
- getEInvoiceOwners

Name	Resource	Method	Description
getEInvoiceOwners	/ Owners	GET	This method answers with a list of all owners of eInvoices stored in the DB

Example	GET http://95.211.XXX.XXX:8080/exchange_server/exchange/fentec/eInvoice/Owners	
Request	{ "customerId": ["Michael] }	
Answer	200 OK	

Figure 6. Get invoice owners method

- getEInvoicesByOwner

Name	Resource	Method	Description
geteInvoiceByowner	/einvoice/[name]	GET	This method returns a list with all the eInvoices of a customer (owner)
Example	GET http://95.211.XXX.XXX:8080/exchange_server/exchange/fentec/eInvoice/ Michael		
Request			
Answer	200 OK	{ "eInvoiceId": ["1839082338", "396673936", "672645074", "676946955", "1956724579"] }	

Figure 7. Get invoices by owner method

- auditEInvoice

Name	Resource	Method	Description
auditEInvoice	/audit/[audit_name]/[invoice_number]	GET	This method returns a piece of an eInvoice according to the privacy policy used to cypher the eInvoice and the set of attributes of the auditor.

Example	GET http://95.211.XXX.XXX:8080/exchange_server/exchange/fentec/audit/taxOffice/1359596144	
Request		

Answer	200 OK	{"taxes": {
		"total paid": 20,
		"tax": 4.2
		}
		}

Figure 8. Audit method

2.3.3 Merchant server

This maven project implements the operation of the Merchant entity, providing a REST API dedicated to the internal platform interwork for purchases. It also provides information regarding products it sells.

2.3.3.1 purchase management

- getContract: this method is invoked by the user application to obtains a contract proposal for a list of products.

- econtractSendBack: this method is invoked by the user application to send the contract signed with eCoins used to pay, and containing all required information for the payment validations, such as validation tokens of each eCoin.

2.3.3.2 Products information

- getMerchantList

Name	Resource	Method	Description
getMerchantList	/merchants	GET	this method is used to obtain a list of all merchant's names.
Example	GET http://95.211.XXX.XXX:8080merchant_server/merchant/fentec/merchants		
Request			

Answer	200 OK	{"merchant": ["FakeMerchant", "BabyCare", "EMT", "Take&Eat"]}

Figure 9. Get list of merchant's method

- getProductsList

Name	Resource	Method	Description
getProductList	/[name]/product List	GET	This method is used to obtain a list of products a merchant sells.
Example	GET http://95.211.XXX.XXX:8080merchant_server/merchant/fentec/EMT/productList		
Request			
Answer	200 OK	{"products": ["Train ticket: 10", "Bus ticket: 10", "Boat ticket: 20"]}	

Figure 10. Get product list method

2.3.4 Trusted Authority

This maven project implements the Trusted Authority entity, that performs all operations related to the management of Functional Encryption crypto schemes and provides the means to link customers, eCoins and purchase operations.

- createAccount: this method creates an account for a user, this can be a customer or a merchant. This operation entails the creation of all crypto keys needed for their operation at Functional Encryption level.

- createECoin metadata: this method is used to create a new type of eCoin into the platform.

- getValidationToken: this method is used to request a validation token for a new eCoin.

2.4 Dependencies

Beyond source code dependencies, the current pilot present external dependencies for their operation.

- Execution environment: The pilot has been programmed to run into a TOMCAT server. Test have been performed in Apache Tomcat v8.5.5x1

- Storage: The storage for contracts and invoices is performed by a MONGO DB2

- Functional Encryption schemes: this pilot make use of the ABE libraries developed in FENTEC project in GOLANG programming language. To entail this integration, we have developed two wrappers which have been published in the FENTEC project repository in Github: https://github.com/fentec-project.

- This integration is performed by building the ABE libraries and GOLANG wrappers as Shared Object Libraries and invoking them from Java with Java Native Access (JNA). Although the common library already contains the JNA integration, it is required to download and compile ABE library and then build SO libraries. Please follow the instructions provided at https://github.com/fentec-project/abe-wrappers

2.5 Deployment and operation

Please visit each project of the pilot in the public repository for further instructions:
https://scm.atosresearch.eu/fentec_digcur_uc_pub

[1] https://tomcat.apache.org/download-80.cgi

[2] https://docs.mongodb.com/manual/release-notes/4.2/

3 Conclusions

This is the final deliverable of the privacy enhanced digital currency prototype which reflects the current status of the pilot fulfilling the requirements of the FENTEC project. This pilot serves to demonstrate how Functional Encryption and ABE technologies provide a new approach to the issue of managing access to private data.

ABE crypto libraries developed in the project, beyond security considerations which are addressed in other work packages, provide an easy way to integrate an elegant solution that allowed to modify and improve the Anonymous cryptographic electronic money protocol developed by David Chaum[4], providing a new protocol to fix issues like the anonymity of customers in case of fraud.

References

[1] FENTEC. D3.1 Technical requirement report analysis. Technical report. European Commission, 2018

[2] FENTEC. D7.5 First version of the privacy enhance digital currency prototype. European Commission, 2019

[3] FENTEC. D7.1 preliminary specification of FENTEC prototypes. Technical report, European Commission, 2018

[4] “Blind Signatures for Untraceable Payments”, Advances in Cryptology: Proceedings of CRYPTO California, 1982