D7.5 First Version of the Privacy Enhanced Digital Currency Prototype

Document Identification			
Status	Final	**Due Date**	31/10/2019
Version	1.0	**Submission Date**	31/10/2019

Related WP	WP7	**Document Reference**	D7.5
Related Deliverable(s)	D7.1, D7.2, D7.3, D7.7	**Dissemination Level(*)**	PU
Lead Participant	ATOS	**Lead Author**	Miguel Angel Mateo (ATOS)
Contributors	ATOS	**Reviewers**	Yolan Romailler (KUD) Clément Gentilucci (FUAS)

Keywords:
Preliminary 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.

(*) Dissemination level.-PU: Public, fully open, e.g. web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Document Information

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

Document History			
Version	Date	Change editors	Changes
0.1	30/07/2019	WALLIX	Published as templates
0.2	11/10/2019	ATOS	First content
0.3	21/10/2019	ATOS	Ready for internal review
0.4	29/10/2019	ATOS	FUAS comments addressed
0.5	29/10/2019	ATOS	KUD comments addressed
0.6	29/10/2019	WALLIX	Final internal review
1.0	31/10/2019	ATOS	Submitted version

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable Leader	Miguel Angel Mateo (ATOS)	21/10/2019
Technical Manager	Michel Abdalla (ENS)	31/10/2019
Quality Manager	Diego Esteban (ATOS)	31/10/2019
Project Coordinator	Francisco Gala (ATOS)	31/10/2019

Table of Contents

List of Figures

List of Acronyms

Acronym	Description
ABE	Attribute-Based Encryption
API	Application Programming Interface
CP-ABE	Ciphertext Policy Attribute-Based Encryption
FE	Functional Encryption
KP-ABE	Key Policy Attribute-Based Encryption
MSP	Monotone Span Program

Executive Summary

This deliverable describes the ATOS privacy-enhanced digital currency use case prototype at the end of the second year of the project. It describes the cryptographic protocol related to Functional Encryption schemes and how they integrate into a payment platform based on blind signature cryptographic scheme. This document presents the first version of the pilot which has been developed as a proof of concept to check the viability of the design. The version presented should serve as base to be extended to fulfil the objectives of the use case.

1 Introduction

Following the Requirements Analysis D3.1 [2] and the Initial Technical Specification D7.1 [4] we now present the initial work into the development of the privacy-enhanced digital currency use case prototype. The goals of this document are to describe, for the ATOS use case:

- how the prerequisites for prototype development have been met,
- the current state of development of the use case in terms of functionality, and
- a preliminary analysis of the security and performance of the prototype.

The Initial Technical Specification D7.1 [4] defined the required functionality for the prototype and how to achieve that using existing tools plus cryptographic protocols defined by the academic partners and the FENTEC library implemented by XLAB. There are also some tentative performance targets but since these proved to be difficult to quantify they are only general guidelines.
The overall goal of this phase of the project (Task 7.2) is to produce a working prototype in order to assess the suitability and efficacy of the cryptographic protocol and its ability to meet the requirements of the application. To this end, in this document, we concentrate on describing how our work meets these requirements. This will then lead into the final phase of the project where we concentrate more upon the performance and security aspects of the use case while maintaining the functional requirements.
This deliverable is produced concurrently with the development of the prototypes for the other two use cases and leads into the reports on the final prototypes plus reports on performance and security:

- D7.3 First version of the truly anonymous data collection prototype (M22)
- D7.4 Final version of the truly anonymous data collection prototype (M33)
- D7.6 Final version of the privacy enhanced digital currency prototype (M33)
- D7.7 First version of the IoT video surveillance prototype (M22)
- D7.8 Final version of the IoT video surveillance prototype (M33)
- D7.9 First test report of the FENTEC prototypes (M23)
- D7.11 Performance report for FENTEC prototypes after first cycle (M24)

1.1 Structure of the Document

section 2.2 describes the cryptographic protocol used for the prototype.

section 2.3 gives a brief description of the platform which has been developed.

section 2.4 outlines the structure of the software and the method of operation.

section 2.5 summarizes what the demonstrator version of the use case application attempts to prove.

section 3 concludes with a brief statement about conformance with project goals.

section 4 gives some brief indications for the next steps in the project.

2 Privacy-Enhanced Digital Currency Demonstrator

2.1 Introduction

The privacy-enhanced digital currency use case aims to provide a new method of payment in order to fulfil three main objectives; to keep the privacy of payers which is also known as unlinkability, prevent fraud, and provide a mechanism to audit the use of the money over the developed system.

To fulfil these objectives we propose to take as a base method, the blind signature scheme developed by David Chaum founded on the principle of untraceable payments [1]. Although at the beginning of the work we proposed to use an existing payment platform, we eventually decided to start with a basic implementation of a new payment platform. This platform does not provide all the characteristics of a mature platform, although it will provide all the necessary mechanisms to meet and demonstrate the achievement of our objectives.
For instance the current implementation does not implement the functionality of giving back cash as it does not require modification of the Functional Encryption part of the demonstrator.
As stated in D4.1 [3], we also consider the convenience of improving the payment platform by adding a new functionality to create coins for specific purposes, this then also becomes an objective of the use case.

The blind signature scheme fulfils our first objective of privacy while, at the same time, offers a way to commit fraud or other crimes. To solve this issue we consider that some information can be leaked to a trusted and independent entity. This information should not compromise the privacy of the users but should be enough to identify them if needed. We defined in D7.1 [4] the role of Issuer of Trust with similar characteristics, trusted and independent and should be provided by a governmental authority independent of the Exchange server. Therefore, the Issuer of Trust will play the roles of FE Master Authority and Privacy provider, while offering a way to revoke this privacy.

Finally, to fulfil the objective of auditability of invoices, we will follow a similar approach to the one presented in the demo of Selective-Access-to-Clinical-Data included in D6.2 [5]. This functionality will be added to the final version of the use case.

2.2 Cryptographic Protocol

In the following sections we simplify the mathematical language in the descriptions of the cryptographic schemes and flows, in order to make the text more interesting to non-crypto experts. People interested in more scientific definitions can find them in the papers cited. We first describe the cryptographic protocol of the blind signature platform itself, in order to achieve a better understanding of how FE schemes integrate into the flow.

As said before, the blind signature scheme is based on Chaum [1] and has been implemented with RSA [7]. It can be described as a system with at least two participants; user and signer, in which the user requests from the signer, and obtains, a valid electronic signature of a message without revealing the content of the message to the signer. This scheme has four main steps:

- **Blinding:** User has a message M and uses a public key PK from the signer and a random factor r to blind the message obtaining:

$$M' = \mathrm{B}(M, PK, r)$$

- **Signing:** User sends M' to the Signer who builds a blinded version of the signature which is sent back to the User. SK is the secret key of the signer.

$$s' = \mathrm{S}(M', SK) = \mathrm{S}(B(M, PK, r), SK)$$

- **Unblinding:** User unblinds s' obtaining a valid signature s of M.

$$s = \mathrm{U}(s', PK, r) = \mathrm{U}(\mathrm{S}(\mathrm{B}(M, PK, r), SK), PK, r)$$

- **Verification:** Later a verifier can easily check that the signature is valid $\mathrm{V}(s, PK)$ where PK is the public key of the signer.

ABE for Fine-Grained Access Control of Encrypted Data [6] is the FE scheme we will use to manage creation an use of eCoins. The implementation of this scheme into the FENTEC project[1] defines a flow with four steps:

- **Setup:** This step performs the creation of a master key MK and public key PK_{fe}, taking as input the universe of attributes UoA considered for each scenario. We use "fe" sub-index to differentiate between this PK and the PK of the blind signature scheme.

$$PK_{fe}, MK = \mathrm{SetUp}(UoA)$$

- **Encryption:** Encrypt a message M to obtain a cyphertext C using PK_{fe} and a subset of the universe of attributes γ, this is a sub-set of attributes we want to associate to M.

$$C = \mathrm{E}(M, PK_{fe}, \gamma)$$

- **Key generation:** This step consists of two operations. First, keys for decryption are generated taking as input a master key MK and an access structure P which is a boolean expression that defines a policy into γ, this access structure has the format of a Monotone Span Program (MSP).

$$KeySet = \mathrm{KGen}(MK, P)$$

  Then, it produces a set of keys for each user taking as input a subset of attributes $attr$ of the user for which it is generated:

$$SK_{fe} = \mathrm{Delegation}(KeySet, attr)$$

- **Decryption:** In this step the cyphertext C is decrypted to obtain M using SK_{fe}, this means that decryption is only possible if the policy P can be satisfied with $attr$ into the set γ:

$$M = \mathrm{D}(C, SK_{fe})$$

Moving this to common language we could say that we will be able to decrypt a cyphertext if the attributes of the key owner $attr$ are enough to satisfy a boolean expression P into the set of attributes γ associated with the message. We will see later how this fits in our use case.

From the point of view of the implementation the decryption step has three possible outcomes:

- $attr$ or P are defined out of γ: the decryption step returns an unreadable output.
- $attr$ or P are defined inside of γ and $attr$ is not enough to fulfil P: the decryption step returns empty.
- $attr$ or P are defined inside of γ and $attr$ is enough to fulfil P: the decryption step returns M.

[1] https://github.com/fentec-project/gofe/blob/master/abe/gpsw.go

2.3 Platform

Currently, the demonstrator consists of a monolithic application which has all required players and entities organized in a set of Java packages and classes to ease the task of building each of the players as independent applications in a second phase of the development of the use case. The application uses the GOFE/ABE[2] library developed in Golang that has been compiled in a Shared Object Library format to be used in Linux and DLL format for Windows platforms. The binding of these libraries with the Java application has been performed with Java Native Access[3].

2.4 Software

The demonstrator has four main entities; Issuer of Trust, Exchange, Customer, Merchant. Each of them is coded into a Java Class and an interface, this interface contains the set of methods that will become the API of the entity in the second phase of the use case (Figure 1).

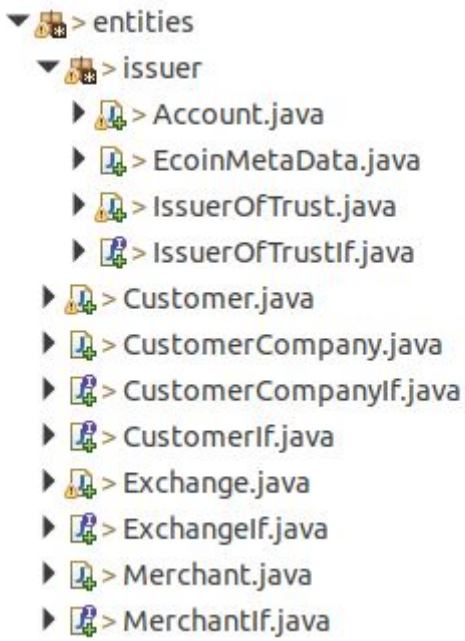

Figure 1: Use case entities

A second package contains classes which represent the required data structures to support the digital cash instantiation and flows (Figure 2).
Finally, two java classes have been implemented to bind the Java code with the Shared Object Library of the Functional Encryption schemes (Figure 3).
This demonstrator can be run in two different ways, one command line interface (Figure 4) and through a Graphic Interface.
The source code of the demonstrator can be found at: `https://github.com/fentec-project`.

[2]https://github.com/fentec-project/gofe/tree/master/abe
[3]https://github.com/java-native-access/jna

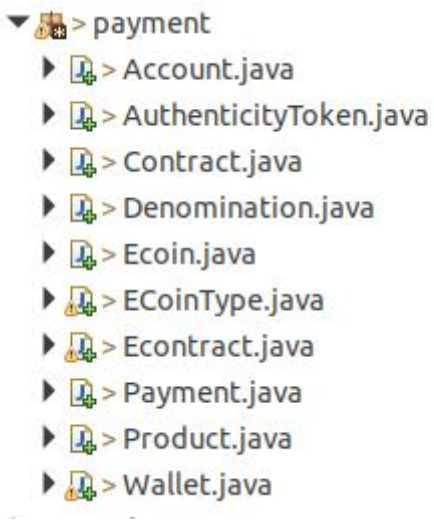

Figure 2: Digital cash classes

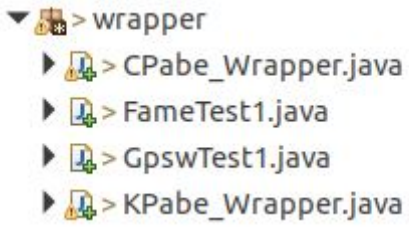

Figure 3: Golang libraries wrappers

```
---------- THE CUSTOMER'S WALLET ----------
####################################
        Base64-Id (16 bytes): F52ya7APXuJWeZlnIzzT2A==
        Funds: 100
####################################
        E-COIN TYPE 1
                Value: 1
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
        E-COIN TYPE 2
                Value: 2
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
        E-COIN TYPE 3
                Value: 3
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
        E-COIN TYPE 4
                Value: 4
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
        E-COIN TYPE 5
                Value: 5
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
        E-COIN TYPE 6
                Value: 10
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
        E-COIN TYPE 7
                Value: 20
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
        E-COIN TYPE 8
                Value: 50
                ECDSA-KeyPair: Public (TRUE); Private (TRUE)
####################################

                debug: Requesting e-coins withdraw to the Exchange by using blind signatures (e.g., 90$)
blindSig: 76602b3484e5424c87d881e72d867ef60ccd50c9b33a6ed87d3d9af3e080a03ba3fc124212fee2aa0b198afe1e7ec10e05edf2201d8e642cdf5e88d515d14cd65
encodedBlindSignature: 76602b3484e5424c87d881e72d867ef60ccd50c9b33a6ed87d3d9af3e080a03ba3fc124212fee2aa0b198afe1e7ec10e05edf2201d8e642cdf5e
blindSig: 20de69c00edefc2615e0043061f2181b6cd306488fb558ca43953481c18707cae072915863a139de5cc0a89a0ffbfa88d9792cce08be5645cca2c70a47cddabd3
encodedBlindSignature: 20de69c00edefc2615e0043061f2181b6cd306488fb558ca43953481c18707cae072915863a139de5cc0a89a0ffbfa88d9792cce08be5645cca2
blindSig: 05e448f7c76384a20173983b3db1a970f78612c7963208cb9fe386d07b2e8d6b310ad40800dc4c28f0907cd702dc3725f495f78c5fa91a04dce4987ff0e8882cd
encodedBlindSignature: 05e448f7c76384a20173983b3db1a970f78612c7963208cb9fe386d07b2e8d6b310ad40800dc4c28f0907cd702dc3725f495f78c5fa91a04dce4
                debug: E-coins requested have been properly signed by the Exchange with the corresponding e-coin type's private key: true
```

Figure 4: Text mode.

2.5 Demonstrator

In this Section we describe through an example how the KP-ABE scheme is integrated into the Chaum scheme to preserve the privacy of customers while adding the mechanisms to avoid some risks inherent to anonymity. At the same time, FE enables authorities to control how money is used.
The following bullets describe some concepts of the use case implementation.

- eCoinMetaData: this is the definition from the point of view of the Issuer of Trust of a type of digital currency, it contains the name of the currency, and the data related to the FE scheme employed to manage its use: universe of attributes, policies, etc.
- eCoinType: this is the instantiation of a type of digital currency in the payment platform. It is a data structure to store specific data to this instantiation like name, key pairs for electronic signature.
- Denomination: this is the definition for each of the coins as a type of digital currency and represents the value of each coin and key pair when used as an electronic signature.
- eCoin: this is the instantiation of a denomination. It is the coin we use to pay and has an identification and a valid signature of this identification.

The flow of eCoins in the demonstrator is explained following a story in which a company gives a daily subvention for lunch to their employees. The help is 4,5 euros per working day, that is 90 Euros per month, which can be only spent in restaurants during a period of three months, after this time this money is no longer valid. Employees receive this monetary help monthly, that is, 90 euros each month. The company contacts the payment platform to manage this subvention. We set the starting date to November 2019.

2.5.1 Instantiation of scenario

The first step is the creation of a new type of currency with its conditions of use. The company requests from the Issuer of Trust the creation of a coin with the following characteristics:

- A new digital cash named "Kudo" with an exchange of 1 to 1 against Euro.
- A set of parameters to characterize Kudos. This corresponds to the **universe of attributes (1 to 30)** of the FE ABE scheme with the following meaning:
  - 1 to 12: months of the year
  - 13: current year (2019)
  - 14: next year (2020)
  - 15 to 19: type of product or business (15: restaurant, 16: gas station, 17: nursery, ...)
  - 20 to 30: type of key owner (..., 29: customer, 30: merchant)
- A set of conditions to apply when this digital cash is used, they correspond with the **policies** of the FE ABE scheme. The boolean expressions of these policies are:
  - policy 1: a customer can use this money in-between the three months after it was issued: P_c=((((11 OR 12) AND 13) OR (1 AND 14)) AND 29)
  - policy 2: a merchant can accept this money as valid if the reception is in-between the three months after it was issued and it is a restaurant: P_m=((((11 OR 12) AND 13) OR (1 AND 14)) AND 15 AND 30)

The second step is the creation of new eCoinType for Kudos at the Issuer of Trust and in the Exchange server. In the Issuer of Trust this step correponds to the setup of the FE scheme, specifically the generation PK_{fe} and MK. In the Exchange server this step includes the creation of RSA/ECDSA key pairs for the eCoinType itself and for each of the denominations of the eCoinType according to the blind signature scheme. These key pairs and PK_{fe} are stored in the Exchange server. they will be used later each time the Exchange issues (blind signing) a new eCoin. Key pairs for the blind signature payment platform are:

- PK_{fe} and MK, at this time it is also possible to generate the decryption $KeySet$ for each of the policies to apply.
- a key pair PK_{ect}, SK_{ect} for the eCoinType
- a key pair for each denomination (5, 10, 20 and 50 Kudos): (PK_5, SK_5), (PK_{10}, SK_{10}), (PK_{20}, SK_{20}), (PK_{50}, SK_{50})

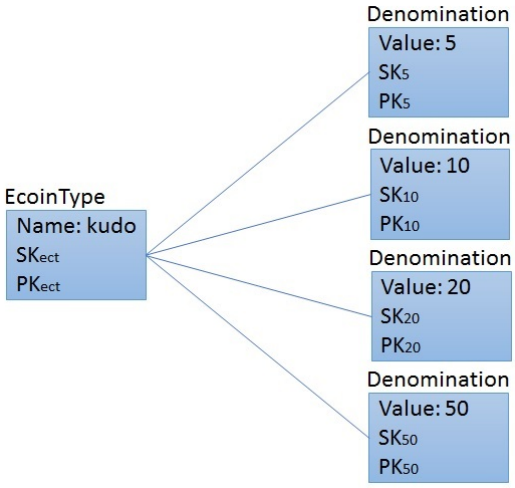

Figure 5: Ecoin type.

The third step is the creation of accounts in the Issuer of Trust and Exchange for employees of the company and restaurants. Customer accounts for employees and merchant accounts for restaurants, all of them work with an electronic wallet application.

- The system creates an account at the Issuer of Trust to obtain a valid identification. Issuer of Trust creates identifications ID_c for employees or customers and ID_m for restaurants or merchants. At this time the Issuer of Trust characterizes each user, customer and merchant, with a set of attributes $attr$ which are used to delegate the keys SK_{fe} needed to use Kudos. These IDs are only known by users and the Issuer of Trust.
- The user initializes an eWallet by creating a random identification (ID) for the wallet, ID_{cw} in the wallet of the customer and ID_{mw} in the wallet of the merchant. These identifications are the only reference known by the Exchange server about customers in order to preserve their privacy. Once the wallet has been identified by the Exchange, it sends Denomination values and public keys to the wallet. This data will be needed later to create eCoins.
- Issuer of Trust generates SK_{feC} and SK_{feM} for Kudos with the specific attributes of each user. In the case of a customers the policy will be P_c and their attributes $attr = (11, 13, 29)$, in the case of a merchant P_m and $attr = (11, 13, 15, 30)$.

- Each wallet has an ECDSA key pair to sign all documents they produce. (PK_c, SK_c) for the wallet of the customer and (PK_m, SK_m) for the wallet of the merchant.

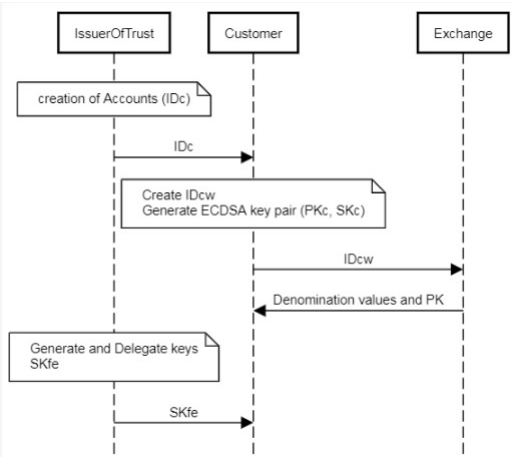

Figure 6: Accounts creation.

2.5.2 eCoin Creation

This step corresponds to the signature step in the Chaum scheme, where the customer creates and blinds an ID for the coin and sends it to the exchange to obtain a blinded signature. It has been modified in order to fulfil the objectives of the use case.

- The company transfers 90 euros to the Exchange for an account identified by ID_c, those 90 euros can be only exchanged for Kudos. If there is no account for that ID in the exchange it is created and the funds are transferred.
- Wallet receives a notification about new funds in its account.
- Customer decides to exchange 50 euros for Kudos.
- Wallet retrieves Denominations of Kudos, these are values and public keys for each denomination.
- Wallet requests a validation ID for the eCoin to the Issuer of Trust.
- Issuer of Trust creates a random validation id ID_v for the eCoin, creates a digital signature of the validation ID with its SK_{iot}.

$$S_v = S(ID_v, SK_{iot})$$

- Encrypts the signature with PK_{fe} and a set of attributes which define the validity of the coin and who can decrypt it, and sends back to wallet the validation ID and the encrypted signature C_v.

$$C_v = E(S_v, PK_{fe}, \gamma); \gamma = (1, 11, 12, 13, 14, 15, 30)$$

- Wallet creates a random ID_r and builds the ID of the eCoin with the function F. This function must have an inverse function F^{-1} to let the Exchange extract ID_v later.

$$ID_{ecoin} = F(ID_v, ID_r)$$

- Wallet creates an ECDSA key pair for the eCoin (PK_{ecoin}, SK_{ecoin})
- Wallet blinds the ID_{ecoin} with the public key of the corresponding denomination, in this case 50 Kudos, and requests a signature from the Exchange.

$$ID'_{ecoin} = B(ID_{ecoin}, PK_{50}, r)$$

- Exchange signs the blinded id and encrypts the signature with PK_{fe} and γ, then the Exchange sends it back to the wallet.

$$S'_{ecoin} = S(ID'_{ecoin}, SK_{50})$$

$$C_{ecoinS'} = E(S'_{ecoin}, PK_{fe}, \gamma); \gamma = (1, 11, 12, 13, 14, 29)$$

- Wallet decrypts the cyphertext of the blinded signature with SK_{feC}. If the policy matches the attributes and the γ used to encrypt it, then the signature is unblinded with PK_{50} and r.

$$S'_{ecoin} = D(C_{ecoinS'}, SK_{feC})$$

$$S_{ecoin} = U(S'_{ecoin}, PK_{50}, r)$$

The wallet will be able to obtain a valid signature for the created eCoin if it is able to decrypt the FE cyphertext, this means that the applied policy is fulfilled with its attributes in γ.
Once the creation of the eCoin has been carried out the eCoin consists of:

- A unique id formed by two different parts, one from the wallet and one from the Issuer of Trust: ID_{ecoin}
- A signature for ID_{eoin}: S_{ecoin}
- A FE cyphertext containing a signature of a part of the id C_v ciphered with $\gamma = (1, 11, 12, 13, 14, 15, 30)$. This cyphertext cannot be decrypted by the wallet as γ does not have attribute 29 required by the policy of the SK_{feC}.
- A value: 50
- An ECDSA key pair: (PK_{ecoin}, SK_{ecoin})

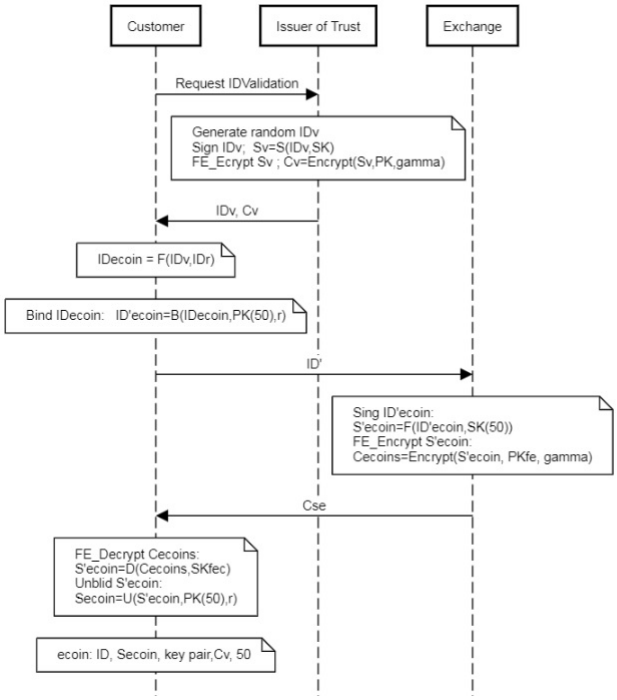

Figure 7: eCoin creation.

2.5.3 Payment

In this step the customer takes a lunch with a price of 50 euros and tries to pay with his eWallet.

- Wallet of the customer requests a contract to the merchant application.
- Merchant creates a contract, signs it with its SK_m and sends it to the wallet of the customer.
- Wallet takes the eCoin with a value of 50 Kudos created before and signs the contract with the private key of the eCoin SK_{ecoin}.
- Add the eCoin with all its data except the private key PK_{ecoin} to the contract and sends it back to the merchant.

2.5.4 Verification

At this step the Merchant verifies the signatures and eCoins to accept them as payment.

- Merchant verifies the signature of the contract with the eCoin ECDSA key.
- Merchant decrypts the FE cyphertext with its SK_{feC} obtaining the signature S_v of the validation ID ID_v which is unknown to him.
- Merchant adds this signature to the eCoin data and sends it to the Exchange server requesting that it be redeemed to his account.
- Exchange extracts ID_v from the id of the eCoin ID_{ecoin}, and verifies that the signature S_v is valid.
- Exchange verifies the validity of the signature S_{ecoin}.

- Exchange checks that no eCoin with id ID_{ecoin} has been used before.
- If all the checks are valid then the exchange transfers 50 euros to the account of the merchant.

At this point we should mention that for the system to work properly over time, keys related to the FE scheme have to be revocated and renewed periodically according to policies.

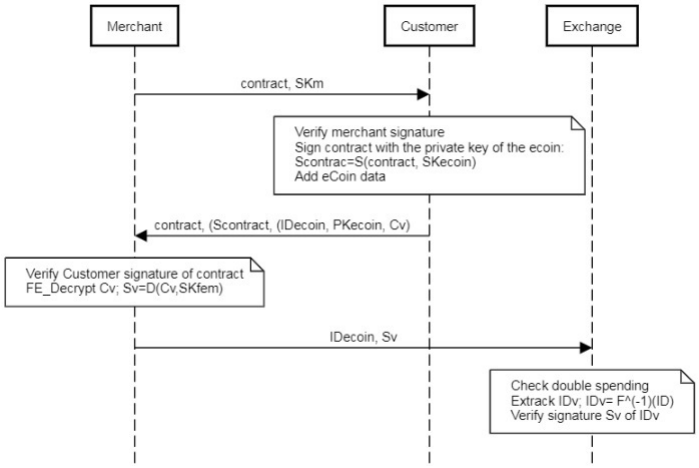

Figure 8: Payment and verification.

3 Conclusion

In this document, we presented the first release of the privacy-enhanced digital currency use case. The release presented in this deliverable integrates the FE KP-ABE scheme with a blind signature encryption scheme to build a payment platform which, while following the principles of Chaum [1], adds the role of a trusted entity, the Issuer of Trust, to manage how money can be spent and solves the issue introduced by anonymity in Chaum's definition. The use of the coin is controlled through the definition of three structures: a set of attributes to describe the characteristics of the money, a policy to define conditions or constraints and a set of attributes to describe the user. Only if those three structures (γ, Policies and $attr$) match can the money be used. This is performed at two different points in the flow, the creation of a new eCoin and the payment with the eCoin. In order to improve the security of the whole scheme, the ID of the eCoin is built with two parameters created by different entities, signed by them and encrypted in a way that enables the identification of an eCoin by the customer who requested its creation. In order to preserve the privacy of customers in this scheme the Issuer of Trust and Exchange roles have to be played by different entities, more precisely, the Issuer of Trust role should be played by a trusted entity such as a governmental organization.

4 Next steps

In order to fulfil the objectives of the use case the demonstrator needs to be enriched with an auditing functionality and also with mechanisms required to enable proper key management. The next step towards FENTEC goals will consist of splitting the current demonstrator to build four applications, one per entity (Customer, Merchant, Exchange, IssuerOfTrust). This change itself entails the integration of a key management system for all keys related to the FE schemes. Together with this new functionality we will integrate a CP-ABE FE scheme to create the Auditing functionality. This scheme is to restrict access to historic data such as invoices or balances by expressing the rules and constraints using FE policies.

References

[1] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 199–203. Plenum Press, New York, 1982.

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

[3] FENTEC. D4.1 annual report on functional encryption schemes for prototypes y1. Technical report, European Commission, 2018.

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

[5] FENTEC. D6.2 preliminary functional encryption toolset api. Technical report, European Commission, 2019.

[6] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[7] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.