decode

Initial decentralised models for data and identity management: Blockchain and ABC MVPs

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 732546

Project no. 732546

DECODE

DEcentralised Citizens Owned Data Ecosystem

D3.4 Initial Decentralised models for data and identity management Blockchain and ABC MVPs

Version Number: V1.0

Lead beneficiary: RU

Due Date: December 2017

Author(s): Paulus Meessen (RU), Alberto Sonnino (UCL)
Editors and reviewers: Ludovico Boratto (Eurecat), Francesca Bria (IMI), George Danezis (UCL), James Barritt (TW) and Denis Roio (Dyne)

Dissemination level:		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approved by: Francesca Bria, Chief Technology and Digital Innovation Officer, Barcelona City Council (IMI)

Date: 31/12/2017

This report is currently awaiting approval from the EC and cannot be not considered to be a final version.

Contents

1 Introduction

In DECODE, we propose an Attribute Based Cryptography access model in which some or all of the data, some or all of the rules, and even some or all of the credentials may be stored in a distributed manner on a ledger. Whenever data, rules and credentials meet at a DECODE node, we determine whether it may be released. This poses several challenges the DECODE project aims to address, providing answers that are informed by previous research like ABC4trust (FP7-ICT-2009-5, proj. nr. 257782) and Sentinels MobileIDM (IIPVV).

This task considers four questions about the use Attribute Based Cryptography in DECODE.

1. How to securely issue and use credentials which are stored in the distributed ledger?
2. How to exert control over such remotely or rather distributed stored credentials?
3. How to combine attribute-based encryption and attribute based credentials to enforce the secure storage of data on the distributed ledger while allowing the right entities to access data?
4. How to authorize transactions on the distributed ledger using anonymous credentials?

This task is about a decentralized model for Attribute Based Cryptography (ABC) and digital identity management and identifying functional technologies that can be used for the project.

With this first deliverable of the task; *initial Decentralized models for data and identity management: Blockchain and ABC*, we intend to identify relevant areas for research in the domains of attribute based cryptography, identity management and distributed ledgers. An overview of current state of the art technologies is provided and, when available, references to software implementations of these technologies as relevant to DECODE are included.

1.1 Attribute Based Cryptography in Decode

As addressed in [29], the DECODE project intends to leverage Attribute Based Cryptography as a tool to give users control over the attributes. In the DECODE system attributes can contain their personal information. It is intended as both a measure of access control to the data and a means to remain in control over one's personal data in the system.

1.2 Document Outline

Section 2 of this document will describe the functionality, usage and different types of Attribute Based Cryptography. Then sections 3 and 4 will address the specific challenges of Identity Management and Access Control respectively. Sections 5 and 6 discuss a scheme that should allow for distributed issuance of credentials on a distributed system such as Chainspace [5].

2 Technologies

2.1 Attribute Based Cryptography

Attribute based cryptography, is a domain of technology that allows for decentralized asymmetric cryptographic operations to be performed based on attributes.

Attribute based cryptography allows for easy use in a distributed context since the attributes are stored in local context. This often prevents a key-distribution problem.

An *Attribute* [6] is any indivisible piece of personal information that can be described by a bit-string, such as an identifier, a qualification or a property of an entity.

A *Credential* [6] is a cryptographic container of attributes that can provide security assurances for all participants in the system.

Credentials are in general described as the relation between three parties. The *issuer*, that binds the attributes into a verified credential. The *relying party* that wishes to know about one or more attributes from one or more credentials. And the entity of the *user* that owns the credential, and exerts cryptographic control over it.

Attribute Based Credentials (ABC) [13, 16] allow users to selectively disclose the attributes from a verified credential. For example a digital identity card as a credential that allows showing where only a single attribute, such as date-of-birth, is disclosed to the relying party. To (abstractly) design a system for ABC one needs a number of cryptographic building blocks [6]. First a commitment scheme is required that binds the user to a tuple of values, with the possibility to apply a signature on the commitment. Next, ABC-protocols should be developed based on this signature. In order to do issuing, a blinded version of the signature is required. For selective disclosure, the signature should provide a proof of knowledge.

Attribute Based Encryption (ABE) [8, 21, 25, 35] allows users to leverage attributes to encrypt or decrypt data. For example, it allows anyone to encrypt a message to a group of people who share a set of attributes, such as all *pet-owners* in your *neighborhood.* To create an ABE system, one has to first determine a suitable way to convert the attributed based access policy into an access tree. Next, this access tree has to be related to a public-private key-pair, related to the used attributes.

Attribute Based Signatures (ABS) [22, 27] allow users to sign messages using a set of attributes from their credentials. For example, medical doctors can digitally attach their credentials to a prescription or advice without them having to distribute a public key to everyone in a network. Attribute Based Signatures are usually created by appending a non-interactive zero knowledge proof [19] of the possession of attributes to a message.

2.1.1 Implementations

While there are many attribute based identity management systems, there are only very few actively maintained projects.

Microsoft maintains the U-Prove implementation. [1] IBM (Zurich) develops and maintains Idemix [16].[2] IRMA was developed at the Radboud University, originally based on Idemix, and is currently maintained by the Privacy by Design foundation. [3] As of writing this document, IRMA is the most actively maintained implementation of ABC.

While there are a number of implementations of ABE available in code repositories like GitHub, none have seen very widespread use, and compatibility with current credential stores used for U-Prove or Idemix can be lacking [35, Section 5.3].

Attribute Based Signatures is currently developed as a feature in the IRMA project.

2.2 Distributed Ledgers

A distributed ledger, or blockchain, is a particular type of high integrity database.

> “The building block of blockchain is hash pointer. A hash pointer is simply some information (typically called transaction) along with a hash of the information. The hash serves to identify the information, and also allows verification of its integrity. A blockchain is a linked list of hash pointers, where pointers have been replaced with hash pointers. The first block in the chain points to a special block called the

[1] `https://www.microsoft.com/en-us/research/project/u-prove/`

[2] `https://www.zurich.ibm.com/identity_mixer/`

[3] `https://privacybydesign.foundation`

> genesis block. Each block contains a hash of the previous block and information specific to the current block. A key result of iterative hashing is that a block implicitly verifies integrity of the entire blockchain before it. Thus given the current head of the blockchain, any party can independently verify the entire blockchain by generating hashes from the beginning of the chain up until the end. The final hash should match that in the head, otherwise the blockchain has been tampered with. Effectively, blockchain can be thought of as a tamper-evident log where data can be appended to end of the log, and tampering with previous data in the log can be detected." [4]
>
> [...]
>
> "A transaction specifies some operation on one or more previous blocks in the blockchain. The result of this transformation, subject to passing validity and verification tests, is a candidate block for being appended to the blockchain. In other words, a transactions represents a function that, if valid, changes the state of the blockchain." [4]

Blockchain contains consensus mechanisms allowing operations in the presence of faulty or malicious nodes. They are needed to prevent faulty transactions becoming recorded into the ledger. The mechanisms for consensus are a meaningful way to compare different distributed ledgers [7]. In particular since some consensus mechanisms, such as proof-of-work, can have huge environmental costs due to large energy consumption.[4]

Whenever a transaction is processed the nodes in a blockchain will compute a script to verify the integrity of the components in the transaction. Most blockchains have a very limited language for these scripts, but others have purposefully been built Turing complete [37]. These extended scripting languages allow the creation of 'Smart Contracts': code that runs within the blockchain network. Smart Contracts allow for functionality beyond a distributed ledger, and for more interaction with external resources.

2.2.1 Implementations

Since the success of Bitcoin [28], blockchain has become a very popular and fashionable technology. There are a great many implementations of blockchains. However not all are equally suitable for the use in DECODE. The most well known, actively maintained distributed ledgers that allow for smart contracts are Ethereum[5] [37] and IBM's Hyperledger (https://www.hyperledger.org/). The Chainspace[6] project [5] is also relevant to DECODE on account of a design that considers both privacy aspects and performance.

3 Identity Management

3.1 Identity

What is an Identity? In a mathematical sense identity is that which remains the same after an operation is performed. The 'Real world' idea of identity can be quite similar, referring to the idea of sameness, but of course dynamically changing through time. This change can also be affected by external parties.

Within technology we often make use of Digital Identity. Digital identity is a technical model that maps the real world identity of a person or entity to the digital space. A digital identity is a collection of claims about an entity. Digital identities may be formed from the combination

[4]The consensus process of bitcoin mining uses currently more energy than 159 individual countries. (`https://powercompare.co.uk/bitcoin/`, accessed 24/11/2017)

[5]`https://www.ethereum.org/`

[6]`http://chainspace.io/`

of many claims, can refer to multiple individuals, and live on beyond the original owner. These digital identities are often constructed based on personal data of the entities.

DECODE is about giving people ownership of their personal data. The tools offered by DECODE should allow users to transact attributes and personal data; and as such have a digital identity within the network. Privacy according to [3] is '*the Freedom from unreasonable constraints on the construction of one's own identity*'. The privacy to construct a digital identity should be a feature offered through the DECODE tools.

3.2 Identity management

Identity management (IDM) is the mechanism or objects used by entities to manage the claims about their digitial identities.

The architecture of identity management system can be divided in two distinct categories: Network Based and Claim Based identity management [33]. In network based identity management the attributes are stored at the identity provider. The users authorize a request by the relying party for access to the attributes. The relying party can then access the data from the identity provider. In claim based identity management, the identity provider makes claims about the attributes of a user. Those claims then are stored at the user. A relying party can request the user to show the possesion of these claims. The user can then use these claims to directly interact with the relying party, without any addition interaction with the issuer.

Some 'offline' identity management systems are well known, such as government ID Documents and membership cards. Typical network based online identity management systems are iDIN (`https://www.idin.nl/`) by the Dutch Banks, or social networks such as Facebook, Twitter and LinkedIn.

Many of these are centralized to single authorities and rarely have the actual entity in control. Though some initiatives such as Uport.me (`https://www.uport.me/`) or Schluss (`https://www.schluss.org/`) offer more distributed options.

Claim based identity management can often be achieved using Attribute Based Credentials.

3.3 IDM using ABC

ABC, as discussed in section 2, allows for a the collection of claims about a person to be gathered at a central 'wallet' that contains the storage and interface at the user.

As mentioned before, the major implementations of ABCs Uprove [13] and Idemix [16] These have been compared in the ABC4trust project [15]. In the performance, and with the possibility of multi-show unlinkability, the Idemix based implementations of ABCs are preferred.

Of current technologies, Attribute Based Credentials seems the most suitable way to integrate identity management in DECODE.

3.4 IDM and the Blockchain

As per [29], a central component in the transaction of attributes within DECODE will be Smart Contracts. Smart Contracts are part of the identity management in DECODE. Chainspace [5] is 'platform agnostic as to the smart contract language', and therefore, can act as a relying part of an existing ABC system such as IRMA[7]. In addition, we envision that decentralized issuance, as per section 5, is going to be a valuable part of moving the control of identity construction back to the users. However, the implications of the use of tokens, transactions registered in a distributed ledger and other outputs from smart contracts as (secondary or derived) credentials remain to be explored.

[7]`https://github.com/credentials/irma_js`

4 Access Control

DECODE is about giving people ownership of their personal data. This section is about the cryptographic means that can be used in DECODE to keep the ownership - in the sense of deciding on access control - with the users.

According to [30], access control is ‘the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied.’

Access control is often mediated through a trusted party that is responsible for deciding and acting on a request for access. A simple example of access control is the need to show the possession of a secret such as a password to access a certain resource. However, access policies are usually more detailed. There are numerous users, in various roles [32], that need to have different sorts of access [24] to a different resources located in various domains [31]. On top of that, all these properties can change over time.

Access Control often turns into a complex issue. This is why access control will often be a very centralized process. Even when federating access control, the authority often lies within a single trusted party.

A solution can be found by incorporating the access control into the resource using encryption. The possession of the right cryptographic key can be used as a simple method of distributed access control. However, this does not allow for complicated access conditions, nor does it imposes any conditions on what happens with the resource after the data has been decrypted.

DECODE requires an access control mechanism for data and attributes. In particular it should allow the users to exert control over the access to personal data.

Attribute Based Cryptography, together with a distributed system could provide the proper means for access control within DECODE.

4.1 Access Control using Attribute Based Encryption

Cryptography (encryption) could be used as a means of access control. Though this can be very limited in level of fine grained control on the access and rarely provides good means for the revocation of access. In particular it creates a key escrow problem, because the user has to give a key to every user that should have access to the resource.

Shamir tried to solve this using an Identity based cryptosystem [34]. This assumes a public identifier, such as an e-mail address, as a public key, and then has a central authority hand out the corresponding private keys to the owners of the identity. This mitigates the key escrow problem, but still relies on a central trusted party.

Attribute Based Encryption (ABE) builds on this idea, allowing data to be encrypted for users who own a particular combination of attributes. The party that wishes to encrypt the data only has to know the public parameters of the issuers of the credentials and can then encrypt the data with a boolean access condition based on the possession of attributes for a single user. This allows for a very fine grained control for the access condition, prevents users from collaborating to show a combination of credentials, and therefore even support negative statements as part of the condition.

Attribute Based Encryption has two varieties. Key-Policy Attribute-Based Encryption (KP-ABE) [21] where the access control policy is encoded in the encryption key, and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [8] where the access control policy is encoded into the cyphertext.

Both of these approaches rely on a key generation authority as a trusted third party.

4.2 Access control using Attribute Based Credentials

It is also possible to have access control based on attributes and credentials without binding the access tree in the cryptography. A third party can be used to create an access token based on the showing of the proper credentials within an access tree. This tokenizer could be a smart contract, and a verifier for these tokens can be stored in a (public) ledger.

This approach can allow for easier invalidation of the access, when the access policies or trust of the issuer changes. In general, since attribute based cryptography is a distributed technology where the credentials are under the local control of the users, revocation is a problem. Some schemes have been proposed for the fast revocation of Attribute Based Credentials [14, 26]. However, if the access control occurs based on a derived credential, such as a token, we allow for additional methods to revoke access. Though the system then needs to have the ability to revoke or invalidate both credentials and tokens.

5 Construction of a Signature Scheme for distributed issuance of ABC

In this section we introduce our construction of a signature scheme that should allow for distributed, multi authority issuance of credentials, and present its formal definition. After presenting the notations and assumptions, we explain the signature scheme on a clear message m, and then extend the construction to support signing on hidden messages, as well as threshold issuance.

5.1 Background and Notations

The following paragraphs present some backgrounds and notations used in the remaining of the paper.

5.1.1 Bilinear Maps

The aggregate signature scheme works in a bilinear map group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p, with an easily computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The map e has the following properties: (*i*) *Bilinearity*: for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $(x, y) \in \mathbb{F}_p^2$, $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$, and (*ii*) *Non-degeneracy*: for all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, $e(g_1, g_2) \neq 1$. The security of the scheme relies on the difficulty of the Computational co-Diffie-Hellman (co-CDH) problem [10].

5.1.2 Zero-knowledge proofs

The signature scheme on hidden messages presented in section 5.4 uses zero-knowledge proofs to assert knowledge and relations over discrete logarithm values. We represent these zero-knowledge proofs with a notation introduced by Camenisch *et al.* [17], as follows:

$$\text{PK}\{(x, y, \dots) : \text{statements about x, y, } \dots\}$$

The above notation means proving in zero-knowledge that the secret values $(x, y, \dots)$ (all other values are public) satisfy the statements provided after the semicolon.

5.1.3 Hashing onto points

The signature scheme requires a cryptographically secure hash function $H : \mathbb{G}_1 \to \mathbb{G}_1$ (mapping an element of group $\mathbb{G}_1$ to another element of $\mathbb{G}_1$).

5.2 Definitions

In this section we describe the set of algorithms that form a randomizable aggregate signature scheme.

❖ Setup(1^λ): defines the system parameters $params$ with respect to the security parameter λ. These $params$ are publicly available and are assumed to be correctly generated.

❖ KeyGen($params$): from the public $params$, each signer generates its own secret key sk and verification key vk.

❖ Sign(sk, m): generate a signature on a message m using the secret key sk.

❖ AggregateSign(σ_0, σ_1): aggregate two signatures σ_0 and σ_1 into one signature.

❖ AggregateKey(vk_0, vk_1): aggregate the public keys vk_0 and vk_1 of two signers into a single key.

❖ Randomize(σ): randomize the signature σ.

❖ Verify(vk, m, σ): verify the validity of signature σ on the message m using the public key pk.

In the case of blind signatures, we replace the algorithms Sign and Verify by the following potentially interactive protocols:

❖ PrepareBlindSign(m) $\leftrightarrow$ BlindSign(sk, c_m, c, π_s): the user requests a blind signature to a signer by proving him in zero-knowledge the correct formation of the commitment c_m and El-Gamal encryption c of m.

❖ ShowBlindSign(vk, m) $\leftrightarrow$ BlindVerify(κ, σ, π_v): To show possession of a signed attribute, we follow the standard approach where the algorithm ShowBlindSign is a proof of knowledge of the credential.

5.3 Signature on Clear Message

The aggregate signature scheme works in a bilinear map group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p, with a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, as described in section 5.1.1. The secret key is a pair $(x, y) \in \mathbb{F}_p^2$ and the verification key is the triplet $(g_2, g_2^x, g_2^y) \in \mathbb{G}_2^3$. Formally, the scheme is defined by the tuple (Setup, KeyGen, Sign, AggregateSign, AggregateKey, Randomize, Verify).

❖ Setup(1^λ): Choose a bilinear map group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with order p, where p is a λ-bit prime number. Let g_1 be a generator of $\mathbb{G}_1$, and g_2 generator of $\mathbb{G}_2$. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2)$.

❖ KeyGen$(params)$: Choose a secret key $sk = (x, y) \in \mathbb{F}_p^2$. Parse $params$ as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2)$, and publish the verification key $vk = (g_2, \alpha, \beta) = (g_2, g_2^x, g_2^y)$.

❖ Sign(sk, m): Parse sk as (x, y). Compute $h = H(g_1^m)$. Output a signature pair $\sigma = (h, h^{x+my})$.

❖ AggregateSign(σ_0, σ_1): Parse σ_0 as (h, ϵ_0) and σ_1 as (h, ϵ_1). and Output $(h, \epsilon_0\epsilon_1)$.

❖ AggregateKey(vk_0, vk_1): Parse pk_0 as (g_2, α_0, β_0) and pk_1 as (g_2, α_1, β_1). Output $(g_2, \alpha_0\alpha_1, \beta_0\beta_1)$.

❖ Randomize(σ): Parse σ as (h, ϵ). Pick a random $r' \in \mathbb{F}_p$ and output $(h^{r'}, \epsilon^{r'})$.

❖ Verify(vk, m, σ): Parse vk as (g_2, α, β) and σ as (h, ϵ). Accept if $h \neq 1$ and $e(h, \alpha\beta^m) = e(\epsilon, g_2)$.

We will now show the correctness and explain the intuition behind the scheme, which is a combination of the protocols proposed by [36] and [11]. In our scheme, signatures are elements of $\mathbb{G}_1$, while public keys are elements of $\mathbb{G}_2$ .In order to obtain a signature on a clear message $m \in \mathbb{F}_p$, the user submits this same message to n signers. Each signer generates a signature $\sigma_i = h^{x_i + my_i}$, where $h = H(g_1^m)$, and (x_i, y_i) is the private key of the signer i. Then, the user aggregates the n signatures to compute σ:

$$\sigma = \prod_{i=0}^{n-1} \sigma_i = \prod_{i=0}^{n-1} h^{x_i} \left(\prod_{i=0}^{n-1} h^{y_i} \right)^m = h^{x+my}$$

where $x = \sum_{i=0}^{n-1} x_i$ and $y = \sum_{i=0}^{n-1} y_i$. Intuitively, generating h from $h = H(g_1^m)$ is equivalent to building $h = g_1^{r'}$ where $r' \in \mathbb{F}_p$ is unknown by both, signers and users. However, since h is deterministic, every signer can uniquely derive it from m; and the fact that h is unique for each m and unknown to the user prevents forgeries.

To verify the signature, the verifier needs to aggregate the verification keys of the signers to obtain a single verification key vk:

$$vk = (g_2, g_2^x, g_2^y) = (g_2, \prod_{i=0}^{n-1} g_2^{x_i}, \prod_{i=0}^{n-1} g_2^{y_i})$$

Then, when the users sends the tuple (m, σ), the verifier checks the paring of the signature using the aggregated key. Since h is an element of $\mathbb{G}_1$, we can express it as $h = g_1^{r'} \mid r' \in \mathbb{F}_p$. Then, the left-end side of the pairing verification can be expanded as:

$$e(h, \alpha\beta^m) = e(h, g_2^{x+my}) = e(g_1, g_2)^{r'(x+my)}$$

and the right-end side:

$$e(\epsilon, g_2) = e(h^{x+my}, g_2) = e(g_1, g_2)^{r'(x+my)}$$

Therefore, $e(h, \alpha\beta^m) = e(\epsilon, g_2)$, from where the correctness of the Verify algorithm follows.

5.4 Signature on Hidden Message

The scheme on hidden messages is defined by the same set of algorithms presented in section 5.3 but by replacing the Sign and Verify algorithms by two potentially interactive protocols, PrepareBlindSign$(m) \leftrightarrow$ BlindSign(sk, c_m, c, π_s) and ShowBlindSign$(vk, m) \leftrightarrow$ BlindVerify$(\kappa,$ σ, $\pi_v)$.

❖ Setup(1^λ): Choose a bilinear map group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with order p, where p is a λ-bit prime number. Let g_1 and h_1 be generators of $\mathbb{G}_1$, and g_2 generator of $\mathbb{G}_2$. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, h_1, g_2)$.

❖ KeyGen$(params)$: Choose a secret key $sk = (x, y) \in \mathbb{F}_p^2$. Parse $params$ as $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, h_1, g_2)$, and publish the public key $pk = (g_2, \alpha, \beta) = (g_2, g_2^x, g_2^y)$.

❖ PrepareBlindSign(m): Pick a random $o \in \mathbb{F}_p$. Set the commitment $c_m = g_1^m h_1^o$, and $h = H(c_m)$. Generate an El-Gamal keypair $(d, \gamma = g_1^d)$ and compute the El-Gamal encryption $c = Enc(h^m) = (g_1^k, \gamma^k h^m)$. Output (c_m, c, π_s), where π_s is defined by:

$$\begin{aligned} rCl\pi_s \;\; = \;\; & \mathrm{PK}\{(m, k, o) : c_m = g_1^m h_1^o \;\wedge\; c = (g_1^k, \gamma^k h^m) \\ & \wedge\; h = H(c_m)\} \end{aligned}$$

❖ BlindSign(sk, c_m, c, π_s): Parse $sk = (x, y)$ and $c = (a, b)$. Recompute h = H (c_m). Verify the proof π_s. If the proof checks, build $c' = (a^y, h^x b^y)$ and output $\sigma = (h, c')$; otherwise output $\perp$.

❖ AggregateSign(σ_0, σ_1): Parse σ_0 as (h, ϵ_0) and σ_1 as (h, ϵ_1). and Output $(h, \epsilon_0\epsilon_1)$.

❖ AggregateKey(vk_0, vk_1): Parse pk_0 as (g_2, α_0, β_0) and pk_1 as (g_2, α_1, β_1). Output $(g_2, \alpha_0\alpha_1, \beta_0\beta_1)$.

❖ Randomize(σ): Parse σ as (h, ϵ). Pick a random $r' \in \mathbb{F}_p$ and output $(h^{r'}, \epsilon^{r'})$.

❖ ShowBlindSign(vk, m): Parse vk as (g_2, α, β), build $\kappa = \alpha\beta^m$ and output (κ, π_v), where π_v is:

$$rCl\pi_v \;\; = \;\; \mathrm{PK}\{(m) : \kappa = \alpha\beta^m\}$$

❖ BlindVerify(κ, σ, π_v): Parse σ as (h, ϵ). Accept if the proof π_v verifies, $h \neq 1$ and $e(h, \kappa) = e(\epsilon, g_2)$.

Figure 1 illustrates the protocol on hidden messages. To keep a message $m \in \mathbb{F}_p$ hidden from the signer, the user and the signer first execute the ObtainSign$(m) \leftrightarrow$ Sign(sk, c_m, c, π_s) protocol. The user generates a Pedersen commitment $c_m = g_1^m h_1^o$ of the message m with randomness $o \in \mathbb{F}_p$, $g_1, h_1 \in \mathbb{G}_1^2$, and $h = H(c_m)$. Then, the user creates an El-Gamal keypair $(d, \gamma = g_1^d)$ and computes the encryption of h^m as

$$c = Enc(h^m) = (a, b) = (g_1^k, \gamma^k h^m)$$

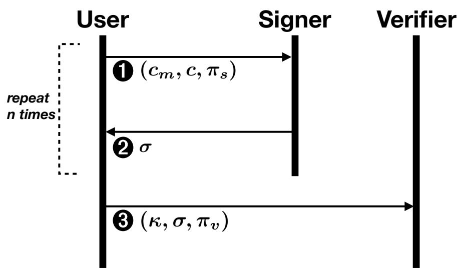

Figure 1: Protocol for blind signatures.

where $k \in \mathbb{F}_p$. The encryption can be randomized by multiplying with an encryption of 0 using a fresh randomness. Finally, the user sends to the signer c, c_m and a proof of knowledge π_s of (m, r, o) ensuring the correctness of the encryption and of the commitment as well as the correct generation of h (❶).

To blindly sign the message, the signer verifies π_s and uses the homomorphic properties of El-Gamal to generate an encryption c' of $h^x(h^m)^y$ as:

$$c' = (a^y, h^x b^y) = (g_1^{ky}, \gamma^{ky} h^{x+my})$$

Upon reception of c', the user decrypts it using his private El-Gamal key d to recover the signature $\sigma = (h, h^{x+ym})$ (❷).

Verification is implemented by the $\mathsf{ShowSign}(vk, m) \leftrightarrow \mathsf{Verify}(h,\ \kappa,\ \sigma,\ \pi_v)$ protocol. The user computes $\kappa = \alpha\beta^m$ from the aggregated verification key, and sends to the verifier (κ, π_v) where π_v is a zero-knowledge proof asserting the correctness of κ (❸). The proof π_v is particularly useful to ensure the signer that the user actually know m and therefore be able to link the attribute m to that user.

5.5 Threshold Aggregation

Our constructions can be modified similarly to [12] to provide a t-out-of-n threshold signature [9] scheme. As the protocol is presented in section 5.3 and section 5.4, the user needs to collect exactly n signatures, one from each signers. A t-out-of-n threshold signature scheme offers more flexibility as the users needs to collect only $t \leq n$ signatures in order to recompute a valid signature (where t is a scheme parameter chosen in advance).

The threshold scheme is similar to the previous ones except for the key generation and the signature aggregation algorithms. For the sake of simplicity, we describe below a key generation algorithm $\mathsf{TTPKeyGen}$ as executed by a trusted third party; this protocol can however be execute in a distributed way as illustrated by Gennaro *et al.* [20].

❖ $\mathsf{TTPKeyGen}(params, t, n)$: Choose two polynomials v, w of degree $t-1$ and set $(x, y) = (v(0), w(0))$. Issue to each signer i a secret key $sk_i = (x_i, y_i) = (v(i), w(i))$ (for each $i \in [1, \ldots, n]$), and publish their public key $vk_i = (g_2, \alpha_i, \beta_i) = (g_2, g_2^{x_i}, g_2^{y_i})$.

The second modification resides in the $\mathsf{AggregateSign}$ algorithm, as the aggregation must be done using the Lagrange basis polynomial l.

❖ $\mathsf{AggregateSign}(\sigma_0, \sigma_1, \ldots, \sigma_t)$: Parse each σ_i as (h, ϵ_i) (for each $i \in [1, \ldots, t]$) and output $(h, \prod_{i=1}^{t} \epsilon_i^{l_i})$, where

$$l_i = \left[\prod_{i=1, j\neq i}^{t} (0-j)\right] \left[\prod_{i=1, j\neq i}^{t} (i-j)\right]^{-1} \mod p$$

Recalling that l allows to reconstruct the original $v(0)$ and $w(0)$ through polynomial interpolation;

$$v(0) = \sum_{i=1}^{t} v(i)l_i \quad \text{and} \quad w(0) = \sum_{i=1}^{t} w(i)l_i$$

one can easily verify the correctness of AggregateSign as below.

$$\begin{aligned} ccch^{x+my} &= h^{v(0)+mw(0)} = \prod_{i=1}^{t} h^{(x_i l_i)} \prod_{i=1}^{t} h^{m(y_i l_i)} \\ &= \prod_{i=1}^{t} (h^{x_i})^{l_i} \prod_{i=1}^{t} (h^{my_i})^{l_i} = \prod_{i=1}^{t} \left(h^{x_i+my_i}\right)^{l_i} \end{aligned}$$

5.6 Remarks

In order to make the signature aggregation work, every signer must operate on the same element h; therefore the algorithm Randomize can only be called after the signature has been aggregated.

Also, our scheme can be easily generalized to sign multiple messages. The signer key pair would become

$$sk = (x, y_0, \ldots, y_{q-1}) \quad \text{and} \quad vk = (g_2, g_2^x, g_2^{y_0}, \ldots, g_2^{y_{q-1}})$$

where q is the number of messages. The multi-message signature will then generalize to

$$\sigma = (h, h^{x+\sum_{j=0}^{q-1} m_j y_j})$$

Interestingly, the size of the signature does not increase with the number of messages, as it is always represented by two groups elements. Then, it is also possible to combine the clear-text and hidden sign protocols to keep only a subset of the messages hidden from the signer, while revealing some others; the BlindSign algorithm will only verify the proof π_s on the hidden messages.

When considering the problem of anonymous credentials for use in a setting where the issuer of credentials is also the verifier, Chasse *et al.* [18], developed a keyed-verification anonymous credentials protocol. This protocol uses a similar bone structure to our scheme and avoids the field paring operation in the verification algorithm by using message authentication codes (MACs).

6 Evaluation of the Scheme

The signature scheme has been implemented in python using the two crypo libraries petlib [1] and bplib [2]. The bilinear pairing works over a Barreto-Naehrig [23] curve, using OpenSSL as the arithmetic backend. We have released the code as an open-source project on GitHub.[8].

Table 1 shows the mean (μ) and standard deviation ($\sqrt{\sigma^2}$) of the execution of each procedure described in section section 5. Each entry is the result of 10,000 measured on an Octa-core Dell desktop computer, 3.6GHz Intel Xeon. This table shows that signing is much faster than verifying signatures (about 15 times faster for the scheme working on clear messages, and 3 time faster for the scheme on hidden messages). Also, aggregation of keys and signatures are extremely efficient.

Table 2 shows the communication complexity and the size of each exchange involved in the signature scheme, as presented in fig. 1. The complexity is expressed as the number of signing authorities (n), and $||m||$ represents the size of the message on which the user wish to obtain a signature. Note that in practice m is the hash of the actual message, and is therefore set to 32 bytes (for SHA-2). The size of a signature is 132 bytes. The highest transaction size appears when the user ask a signature on a hidden message. This comes from the fact that the proof π_s associated with the message is approximately 318 bytes; the proof π_v is only 157 bytes.

[8]https://github.com/asonnino/co-cosign

Number of measures: 10,000

Operation	μ [ms]	$\sqrt{\sigma^2}$ [ms]
Keygen	2.392	± 0.006
Sign	0.445	± 0.001
AggregateSign	0.004	± 0.000
AggregateKeys	0.017	± 0.000
Randomize	0.545	± 0.002
Verify	6.714	± 0.005
PrepareBlindSign	2.633	± 0.003
BlindSign	3.356	± 0.002
ShowBlindSign	1.388	± 0.001
BlindVerify	10.497	± 0.002

Table 1: Performances evaluation.

Number of authorities: n, Signature size: 132 [B]

Transaction	complexity	size [B]
Signature on clear message:		
❶ ask signature	$O(n)$	$\|\|m\|\|$
❷ get signature	$O(n)$	132
❸ verify signature	$O(1)$	$\|\|m\|\| + 132$
Signature on hidden message:		
❶ ask signature	$O(n)$	516
❷ get signature	$O(n)$	132
❸ verify signature	$O(1)$	355

Table 2: Communication complexity and transaction size.

7 Conclusion

This document has outlined the current state of the art in Attribute Based Cryptography and its applications as it relates to DECODE. In particular the applications in identity management and access control.

DECODE is a distributed system of tools for transacting attributes, some of which may contain personal data. In order to give the people ownership of their their data in DECODE, the intent is to combine the privacy enhancing features of Attribute Based Cryptography with the autonomy that comes from using a distributed system.

A scheme for multi authority signatures has been outlined that allows for threshold issuance of credentials in a distributed manner. Implementation of such a scheme within DECODE should allow new credentials to be created within the network without relying on a single trusted partner as identity provider, and without users having to give up ownership of their credentials.

Both Attribute Based Encryption and Attribute Based Signatures are useful for access control and identity management in DECODE, but still lack practical implementations that work well with existing credentials wallets. However, levering the language agnostic features in the Smart Contracts that run in Chainspace, a large part of the functionality might be emulated using Attribute Based Credentials.

The relation between credentials and their secondary or derived cryptographic counterparts, especially those permanently recorded in a public ledger remains for further research, and can contribute to D3.8.

References

[1] petlib. `https://github.com/gdanezis/petlib`, 2017 (version July 20, 2017).

[2] bplib. `https://github.com/gdanezis/bplib`, 2017 (version October 12, 2017).

[3] AGRE, P. E., AND ROTENBERG, M. *Technology and privacy: The new landscape*. Mit Press, 1998.

[4] AL-BASSAM, M., BANO, S., DANEZIS, G., DEVILLIERS, M., AND SONNINO, A. D3.1 survey of technologies for abc, entitlements and blockchains, 2017.

[5] AL-BASSAM, M., SONNINO, A., BANO, S., HRYCYSZYN, D., AND DANEZIS, G. Chainspace: A sharded smart contracts platform. *arXiv preprint arXiv:1708.03778* (2017).

[6] ALPAR, G. *Attribute-based identity management: bridging the cryptographic design of ABCs with the real world*. Uitgever niet vastgesteld, 2015.

[7] BANO, S., SONNINO, A., AL-BASSAM, M., AZOUVI, S., MCCORRY, P., MEIKLEJOHN, S., AND DANEZIS, G. Consensus in the Age of Blockchains. *ArXiv e-prints* (Nov. 2017).

[8] BETHENCOURT, J., SAHAI, A., AND WATERS, B. Ciphertext-policy attribute-based encryption. In *Security and Privacy, 2007. SP'07. IEEE Symposium on* (2007), IEEE, pp. 321–334.

[9] BOLDYREVA, A. Efficient threshold signature, multisignature and blind signature schemes based on the gap-diffie-hellman-group signature scheme. *IACR Cryptology ePrint Archive 2002* (2002), 118.

[10] BONEH, D., AND FRANKLIN, M. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO 2001* (2001), Springer, pp. 213–229.

[11] BONEH, D., GENTRY, C., LYNN, B., AND SHACHAM, H. Aggregate and verifiably encrypted signatures from bilinear maps. In *Eurocrypt* (2003), vol. 2656, Springer, pp. 416–432.

[12] BONEH, D., LYNN, B., AND SHACHAM, H. Short signatures from the weil pairing. *Advances in Cryptology—ASIACRYPT 2001* (2001), 514–532.

[13] BRANDS, S., AND PAQUIN, C. U-prove cryptographic specification v1. *Microsoft Corporation* (2010).

[14] CAMENISCH, J., DRIJVERS, M., AND HAJNY, J. Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society* (2016), ACM, pp. 123–133.

[15] CAMENISCH, J., KRENN, S., LEHMANN, A., MIKKELSEN, G. L., NEVEN, G., AND PEDERSEN, M. Ø. Scientific comparison of abc protocols.

[16] CAMENISCH, J., AND LYSYANSKAYA, A. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Advances in Cryptology—EUROCRYPT 2001* (2001), 93–118.

[17] CAMENISCH, J., AND STADLER, M. Proof systems for general statements about discrete logarithms.

[18] CHASE, M., MEIKLEJOHN, S., AND ZAVERUCHA, G. Algebraic macs and keyed-verification anonymous credentials. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 1205–1216.

[19] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques* (1986), Springer, pp. 186–194.

[20] GENNARO, R., JARECKI, S., KRAWCZYK, H., AND RABIN, T. Secure distributed key generation for discrete-log based cryptosystems. In *Eurocrypt* (1999), vol. 99, Springer, pp. 295–310.

[21] GOYAL, V., PANDEY, O., SAHAI, A., AND WATERS, B. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security* (2006), Acm, pp. 89–98.

[22] HAMPIHOLI, B., ALPÁR, G., VAN DEN BROEK, F., AND JACOBS, B. Towards practical attribute-based signatures. In *International Conference on Security, Privacy, and Applied Cryptography Engineering* (2015), Springer, pp. 310–328.

[23] KASAMATSU, K. Barreto-naehrig curves. `https://tools.ietf.org/id/draft-kasamatsu-bncurves-01.html`, 2014 (version August 14, 2014).

[24] LAMPSON, B. W. Protection. *SIGOPS Oper. Syst. Rev. 8*, 1 (Jan. 1974), 18–24.

[25] LEWKO, A., AND WATERS, B. Decentralizing attribute-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2011), Springer, pp. 568–588.

[26] LUEKS, W., ALPÁR, G., HOEPMAN, J.-H., AND VULLERS, P. Fast revocation of attribute-based credentials for both users and verifiers. *Computers & Security 67*, Supplement C (2017), 308 – 323.

[27] MAJI, H. K., PRABHAKARAN, M., AND ROSULEK, M. *Attribute-Based Signatures*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 376–392.

[28] NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system, 2008.

[29] ROJO, D., BARRITT, J., HOEPMAN, J.-H., DE VILLIERS, M., SAMUEL, P., DANEZIS, G., DEMEYER, T., BANO, S., AND SAGARRA, O. D1.4 decode architecture overview, 2017.

[30] SAMARATI, P., AND DE VIMERCATI, S. C. *Access Control: Policies, Models, and Mechanisms*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 137–196.

[31] SANDHU, R. S. Lattice-based access control models. *Computer 26*, 11 (1993), 9–19.

[32] SANDHU, R. S., COYNE, E. J., FEINSTEIN, H. L., AND YOUMAN, C. E. Role-based access control models. *Computer 29*, 2 (Feb. 1996), 38–47.

[33] SCUDDER, J., AND JØSANG, A. Personal federation control with the identity dashboard. In *IDMAN* (2010), Springer, pp. 85–99.

[34] SHAMIR, A., ET AL. Identity-based cryptosystems and signature schemes. In *Crypto* (1984), vol. 84, Springer, pp. 47–53.

[35] VAN DE KAMP, T. Combining abcs with abe: Privacy-friendly key generation for smart card based attribute-based encryption, August 2014.

[36] WATERS, B. Efficient identity-based encryption without random oracles. In *Eurocrypt* (2005), vol. 3494, Springer, pp. 114–127.

[37] WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper 151* (2014).