Blockchain and Privacy: Use Cases and Architectures*

Working Group on Blockchain and Privacy
Innovation Network for Finance IT

2019/01/07 at 13:19:08

Abstract

This project investigates how blockchain technology can be made consistent with GDPR and provide a platform for auditable GDPR-compliant personal information management. The present report contains concrete use cases in this realm and preliminary architecture designs.

1 Introduction

In this report we examine concrete use cases of blockchain technologies in the context of the GDPR. We focus on two cases: robust consent management between controllers and data subjects (§ **??**), and a system for education diploma management (§ **??**).

For the consent management use case, we demonstrate how blockchain concepts can be used to *facilitate* core GDPR principles, and develop as system prototype which allows controllers and data subjects to maintain a robust log of data usage consent.

Our second use case is inspired by DiplomaSafe, an organization which currently provides a service for issuing and verifying digital diplomas attesting to degree and course completion. We develop a system which allows DiplomaSafe to partially outsource its solution to a blockchain system thus ensuring that their service remains valuable even in the case where DiplomaSafe discontinues operation. We analyze the privacy implications of our solution and provide a legal argument for why our design is compliant with the GDPR, despite the use of a public, permissionless blockchain system.

2 Use Cases

2.1 Consent Management

The GDPR mandates that data controllers implement robust consent management systems and integrate these into their operational setup.

Consent managements tasks include collecting consent from data subjects for specific data processing tasks, allowing data subjects to manage consent (which includes revocation), and facilitating audits (these may be private or public), to demonstrate that all processing of personal data is backed by the active consent from the affected data subjects. One challenge we foresee in consent management is resolving disputes between data subjects (or organizations acting on their behalf)

*Milestone 2 report (draft); for internal distribution only.

and controllers. In particular, a data subject should be empowered to contend that a data processing step taken by a controller occurred without proper consent (*e.g.*, the data subject never consented to the processing, or revoked consent before the processing took place). In this case, a controller must demonstrate that consent was given, or otherwise be subject to fines.

The naive solution of a controller using its own local records to demonstrate consent however raises questions of integrity. In particular, the controller could simply manufacture fraudulent consent records. In this report we explore decentralized architectures that provide more robust solutions to dispute resolution.

2.2 DiplomaSafe Digital Diploma Management

DiplomaSafe, a Copenhagen-based start-up, provides a service for digital diploma management.

In particular, DiplomaSafe allows schools, course providers, and other educational organizations to issue digital certificates to students to attest that the students have completed a particular course or degree.

When a student demonstrates the digital diploma to an interested party (*e.g.*, potential employer), the party can verify the legitimacy of the diploma with DiplomaSafe (which acts as a trusted intermediary in this case). In addition to issuing diplomas and their verification, DiplomaSafe also provides the option to revoke diplomas (in case a diploma has been issued by mistake, or the contents of the diploma are incorrect, *e.g.*, misspelled names, incorrect course information).

One problem an attestation service such as DiplomaSafe faces is assuring potential customers that the service will remain available long enough to be valuable and worth investing in. In a "naive" solution setting, should DiplomaSafe go out of business, much of the service loses its value. In particular, it becomes impossible to verify issued diplomas since DiplomaSafe is the only point of verification.

We investigate how cryptographic tools and blockchain technologies can be leveraged to ensure that all the assets created by DiplomaSafe remain valuable, even if DiplomaSafe discontinues its operation, as well as ways to continue the service without DiplomaSafe's involvement.

In the context of this use case, we define the following entities.

- **Issuer.** An organization such as a school which provides courses to recipients, *i.e.*, students, and issues digital diplomas upon successful completion.
- **Recipient.** A natural person who has completed a course offered by an issuer and holds a digital diploma attesting to this fact.
- **Diploma.** A document issued by an issuer towards a recipient attesting to a qualification, *e.g.*, course completion. A diploma identifies an issuer and a recipient. A diploma can be in an active or a revoked state.
- **Challenger.** An organization or natural person requesting verification of a diploma.
- **Diploma Management Service (DPS).** A service which implements the following functionalities:
  - *Issuing.* Allows an issuer to issue a diploma to a recipient such that the diploma can later be verified.
  - *Verification.* Allows a challenger holding a diploma to verify that the diploma is valid, *i.e.*, issued by the issuer to the recipient as identified on the diploma, and in an active state.

- *Revocation.* Allows an issuer of an active diploma to mark it as revoked.
- *Storage.* Allows a recipient to store and retrieve a diploma issued to them.

- **Facilitator** The entity maintaining and operating the DPS (currently DiplomaSafe).

Our aim is to extend DiplomaSafe's current DPS such that *(i)* all diplomas issued via the DPS can still be verified without direct involvement of DiplomaSafe (*i.e.*, remain useful should DiplomaSafe discontinue its operation), and *(ii)* this is achieved without falling back on burdening the individual issuers with manually verifying each diploma upon a request from a challenger.

To this end, we develop the following architecture.

2.2.1 DiplomaSafe Extended Architecture

In order to make the DPS operable without DiplomaSafe's continued involvement, we leverage a public, permissionless blockchain system, in particular the Ethereum platform.

At a high level, every time a new diploma is issued, the issuer hashes it first and commits the result to the Ethereum platform. Since the transaction is signed by the issuer this serves as an immutable record of the fact that the issuer (as identified by the public key used for the transaction) issued the diploma. To verify the authenticity of a given diploma, a challenger can simply hash it and further verify that the same hash exists on the Ethereum platform. This workflow does not require the active participation of any single facilitator, *i.e.*, DiplomaSafe (barring the initial system setup which we discuss later).

We note that a diploma constitutes personal data and as such falls under the GDPR. Since the hash of each diploma is uploaded to the Ethereum platform it becomes accessible to the public. To ensure that the hash does not leak information about the diploma (via a brute-force attack), the issuer embeds a random nonce in the diploma before hashing (a fresh nonce is generated for *each* issued diploma). Intuitively, this makes it infeasible for an attacker to derive any personal information from a given hash without holding the diploma that produced it. As such, we argue that the hash in isolation constitutes anonymous data and thus falls outside the scope of the GDPR. We further expand on this in Section 2.2.2.

To handle revocation, the hash is augmented with a flag indicating whether it is active or not. To revoke a diploma, an issuer creates a new transaction which updates the flag.

Further we outline a concrete system based on the above ideas.

Prototype System. In our prototype system, the functionality of the DPS is distributed across DiplomaSafe and the Ethereum platform, where DiplomaSafe acts as a facilitator and handles diploma storage and issuer administration (*e.g.*, new issuer registration). DiplomaSafe furthermore provides software for creating well-formatted digital diplomas, which includes generating and embedding a random nonce in each diploma during creation.[1]

An Ethereum contract implements diploma digest upload, verification, and revocation.

The data structures used in the smart contract are as follows.

- **Issuers.** A list of public addresses associated with issuers, along with additional details (*e.g.*, organization name) about each issuer.

[1] By default, DiplomaSafe offers this software as a web service but should it discontinue operation, DiplomaSafe will first distribute the software to all issuers for local deployment and use.

- **Diploma Digests.** A list of digital diploma digests. Each digest is accompanied by a flag indicating whether the diploma is active or revoked, a timestamp, and the public address of the issuer who issued the diploma.

The smart contract provides the following functionality.

- **Issuer Registration.** Each issued diploma is uniquely tied to an issuer via a public Ethereum address associated with the issuer. Before using the system, an issuer creates an Ethereum private key and a public address and registers it with DiplomaSafe. DiplomaSafe adds the issuer's address and associated contact details to the Issuers list.
- **Issuing.** To issue a diploma, an issuer uses the diploma creation service to construct a valid diploma. This generates a string $s = d|n$ where d is the textual representation of the diploma, n is a random nonce, and $|$ denotes string concatenation. The issuer hashes s locally to obtain digest h and uploads h via the smart contract which creates a new timestamped entry in the Diploma Digest list.
- **Verification.** Given a diploma s, a challenger hashes it and verifies via the smart contract that the Diploma Digests list contains the resulting digest with an active flag and the correct issuer.
- **Revocation.** To revoke a diploma with digest h and issuer I, issuer I locates the entry in the Diploma Digests with digest h and updates its flag from active to revoked. The contract enforces that only the original creator of a digest can update its state flag.

As previously noted, the digital diplomas themselves are stored with DiplomaSafe as well as locally with issuers, recipients, and challengers, *i.e.*, off-chain.

2.2.2 GDPR Compliance Discussion

The above use case and architecture center around handling personal data while also leveraging a public, permissionless blockchain system.

As such, it is crucial to analyze the validity, and legality, of the proposed deployment. Currently, there is much contention around blockchain technologies and whether these can be operated in a way that is compliant with the GDPR. In this section we argue that the deployment proposed above is a positive illustration of a GDPR-compliant system, despite its use of a public, permissionless blockchain. To solidify this claim, we tie together legal and technical analyses.

Available Literature and Analysis. We first summarize and discuss available literature pertaining to the deployment. In particular, we include working group opinions and recommendations regarding blockchain technologies and cryptographic techniques and their relation to the GDPR. We focus primarily on cryptographic hashing and data anonymization as these are especially relevant to the DiplomaSafe use case.

In "Blockchain and GDPR" [Blo18], a thematic report prepared by The European Union Blockchain Observatory and Forum, it is stated that hashed personal data is a grey area. It is important to note that, as of yet, this matter has not been settled by the European Data Protection Board (before known as art. 29 Working Party or by any other data protection authorities or in court).

expand on text below

[Art14] discusses anonymization and states that any form of hashing only achieves pseudonymisation.

Cryptographic Hashing and Anonymization under the GDPR. While it is generally accepted, that an immutable public, permissionless blockchain is inherently at odds with the principles set out in relevant data protection legislation and namely the GDPR, it becomes increasingly difficult to imagine the viability of blockchain and distributed ledger technologies, if we are to accept that technology hash functions de facto are viewed only as pseudonymous and therefore subject to GDPR.

When taking the criteria for anonymization into account and with an outset in the DiplomaSafe use case, it is imperative to keep the objective of the rules in mind, namely that of "protecting individuals". As set out in Opinion 4/2007 by the Working Group, it is brought forward that, "The scope of application of the Directive excludes a number of activities, and flexibility is embedded in the text to provide an appropriate legal response to the circumstances at stake" while maintaining that "the scope of the data protection rules should not be overstretched".

A key aspect of our solution is applying a cryptographic hash function to create a digest, i.e., a digital signature, of a diploma, and uploading this digest to a public, permissionless blockchain.

According to the working group's assessment, any form of hashing can at most produce pseudonymous data which would render making the digest publically available a violation of the GDPR.

However, we argue that the carefully tailored usage of a hash function, in the context of this specific use case, provides security guarantees that are in fact as strong as any possible form of generalization or randomization (methods of anonymization), which in turn implies that if anonymization is to be achievable at all, hashing—under certain conditions—must in fact achieve it.

As the working group article points out, mere hashing is not sufficient to protect against inference, since the output of the hash function can be used in a brute-force attack. An attacker attempting to recover a student's personal data could enumerate all realistic guesses at the content of the student's diploma, create validly formatted diplomas from these guesses, and verify if any of these produce the desired hash, thus breaching the student's privacy.

This is why we strengthen the hashing approach by embedding, in each diploma, a sufficiently long random string, which we refer to as a nonce. We emphasize that a new nonce is generated for each diploma. In that sense, the approach is akin to a keyed hash, however each diploma has a unique key and furthermore, the key does not exist separately from the diploma.

This additional security guard makes a brute-force attack infeasible since the attacker would have to guess a (arbitrarily long) random string in order to use the enumeration approach described previously. The computational effort required in this case is exponential in the bit length of the random nonce and as such prohibitively high for sufficiently long nonces.

The only other ways in which an attacker can carry out an inference attack is either by breaking the underlying hash function (in particular, finding a way to invert it) or by obtaining the nonce associated with the diploma. Inverting a cryptographic hash function, when the input domain is sufficiently large, goes far beyond reasonable effort. We note that while it is possible that practical collision attacks might surface for state-of-the-art cryptographic hash functions, these do not lead to inference attacks in our case. An actual inversion is necessary to recover the input. Even for hash functions where practical collision attacks do exist, e.g., MD5, inversion attacks over large input domains do not. As such it is unrealistic to anticipate such attacks on current state-of-the-art hash functions such as SHA256.

The second potential inference attack requires the attacker to first obtain the random nonce embedded in a given diploma. However, by design, the nonce is never stored separately from the diploma itself. Therefore, an attacker can only obtain the

nonce by obtaining a copy of the diploma. This implies that any inference attack on the diploma via the uploaded digest is obsolete since the attacker must already hold the actual diploma. In turn, this means that making the digest of the diploma publically available does not introduce a new attack vector.

It follows then that, in any practical scenario, the digest in isolation (without a valid nonce) does not hold any additional information an attacker can exploit to infer personal data. As such, the digest must constitute anonymous data.

To further substantiate the claim that the above hashing approach offers proper means of anonymization we briefly contrast it with the two candidate approaches to anonymization the working group Opinion 5 article mentions, namely generalization and randomization.

check

generalization and randomization

Overall System Compliance Analysis. In the context of the DiplomaSafe use case, four main points should be kept in mind: (a) the Data Subjects personal data does not appear on the public, permissionless blockchain, (b) there is a Data Controller that is responsible for the Data Subjects information in its original and plain-text state and as such the Data Subject can exercise fundamental rights, such as deletion thus rendering the data anonymous under the current guidelines set forth by the article 29 Working Group (c) any security breach would be traceable to the Data Controller, because this information would be embedded in the diploma and (d) the Data Controllers in question would be continually responsible for ensuring that the technology used is consistent with the technological developments, such as computational effort.

In regard to point (a), the Data Subjects information does not as such appear in the data set. As illustrated by the description of the local key-coded hash and upload process, the only information available on the public, permissionless blockchain is the key-coded hash. In isolation, the key-coded hash is [impossible] to glean a Data Subjects personal data from. It would be improbable that it would even be possible to glean whether there underlying data referenced any real data points or was simply randomly generated.

In regard to point (b), it is important to remember, that the Data Controller(s) – in this case the University or the employer who receives the CV still exists and must uphold the principles set forth in the GDPR, irrespective of the key-coded hash on the public, permissionless blockchain. In addition, if the University as the original key-code hasher, were to respond to a request for erasure under Article 17, the key-coded hash would no longer be technically traceable to the original Data Subject and the underlying information. [The risk of linkability and inference would therefore be severed.]

In regard to point (c), any breach of security would be traceable to the place of breach of this information would be embedded in the diploma. Therefore, the Data Subject would not forego the notification rights set out in the GDPR.

In regard to point (d), assurance is provided that Data Subjects personal information will not thoughtlessly be subjected to the key-code hash process and uploaded to the public, permissionless blockchain without regard for the ever-current state of technology as the Data Controller still is subject to the GDPR and thus must continuously revisit what is "likely reasonably" give an increase in computational power, technological developments and other relevant information.

3 Consent Management Architectures

In this section we describe the architecture of a number of high level systems working towards solving the above described problems. We will consider a very general usecase with just two primary actors: a data subject and a (potential) data controller. Here the data subject gives consent for the data controller to the processing of some personal data of the data subject. The data controller is given access to the personal data which, providing consent was given, the data controller may process. As per the GDPR rules the data subject should be able to revoke consent at which point the data controller should stop processing the data (and delete it). The focus of the systems sketched below will be on resolving any disputes that may arise between the data subject and data controller about whether or not consent for data processing was given and/or revoked.

3.1 Naive Consent Management

As a baseline we outline how we consent management might work in a naive system. In this setting the interaction is strictly between the data subject and the data controller. The data subject gives consent to the data controller and then transfers personal data directly to the data controller for processing. The concrete method used to give consent could vary (e.g., the data subject clicks an "Agree" button on a website). Regardless of the method, the consent given is only logged by the two actors locally. In the case of disputes we must rely on the trustworthiness and accuracy of the logs of the two actors in order to settle the dispute. Keeping and managing such logs is not trivial (in particular for a private person acting as data subject). If the parties fail it may result in inaccurate or lost logs. Furthermore, the parties may tamper with their logs on purpose. This is obviously problematic in the case of disputes as the logging performed by the two parties may well be ambiguous (by change or malintent). I.e., in this setting it may be very difficult for either party to argue their case in a dispute as the argument becomes word against word.

3.2 Consent Management Using a Trusted Third Party

A basic idea to help resolve disputes between the data controller and data subject would be to introduce a trusted third party. We denote this third party the *consent authority*. In such a system, the data subject would register the consent given to the data controller with the consent authority before transferring personal data to the data controller. Similarly, the data subject should use the consent authority to register revocation of consent. In either case it is the job of the consent authority to log the consents given and revoked by data subjects. The data controller should then refer to the consent authority in order to decide whether or not consent for processing was given and/or revoked.

In case of a dispute the consent authority can settle the dispute between the two parties by referring its logs. If the two parties accept the consent authority as the authority on settling disputes this helps as we would no longer have the problem of ambiguous logs.

However, this solution has new disadvantages. The consent authority becomes a single point of trust. I.e., the solution pushes the responsibility of accurately keeping the log of consents to this authority. Failure in keeping the logs will still result in problems when solving disputes and if for some reason the service provided by the consent authority becomes unavailable the whole system will fail to work. Additionally, since the consent authority has essentially full control of the logs,

the parties must trust the consent authority to not tamper with the logs (possibly influenced by the opposing party, e.g., via bribes).

Thus in this system the data subject and controller become highly dependent on a single trusted party as the system is not robust to failures on the part of the consent authority. Finding a trusted third party that the data subject and controller both trust to act as consent authority can be difficult, and compensating the authority for its service may well be costly.

We note that, it may be possible to reduce the required trust in the consent authority using technical means, such as digital signatures and hash chains. However, certain aspects of trust appear difficult to replace. For example, how to protect against a consent authority that simply refuses to accept a consent revocation from a data subject?

3.3 Consent Management Using a Blockchain

To solve the problem in the above system, we could imagine replacing the trusted third party using a blockchain system. In fact, one of the main proposed advantages of blockchain technology is to distribute the trust in a single trusted party (or a few parties) over a network of parties maintaining the blockchain system. Such a system would work similarly to the system using a trusted third party, only the data subject registers his consent as a transaction (possibly a smartcontract?) on a blockchain. Consent can then be said to be given once the transaction is settled on the blockchain. The data controller should refer to the blockchain in order to update is view of the state of consents.

3.4 Case: Consent Management Protocol

Based on the observations made in the previous sections, a consent management protocol is here proposed.

3.4.1 Design Criteria

Here are listed the design criteria deemed necessary for a successful consent management protocol.

Indisputable Consent States

One can see the state of a consent relationship between a controller and a subject as a binary matrix, where each row corresponds to a category of data and each column corresponds to a method of processing. If a bit in the matrix is 1, it means the controller has the consent to use a method of processing corresponding to the column of the bit, on the category of data corresponding to the row of the bit. If the bit is 0, this consent is either not given or has been revoked.

The initial state is all zeros, and the only operation allowed on the matrix is flipping exactly one bit, this operation can only be conducted by the subject.

The protocol should only allow operations such that the view of the matrix is the same from the perspective of the subject and the controller.

A *consent chain* is defined as a chain of operations where the consent matrix can be directly inferred by the state of the chain. It should not be possible for one party to produce a consent chain, which the other party has not explicitly agreed to and they cannot dispute.

Two Party Consent Chain

There are exactly two parties that should know the state of the consents: the

subject and the controller. To ensure this all protocol operations should be either between the subject and the controller, or divulge no information on the state of the consents.

Blockchain Consent Authority
In the case of an unresponsive, reluctant or malicious data controller, it should be possible to have a third party acknowledge consents and objections. It would be preferable to minimise the involvement of the consent authority, as transaction fees might make involvement costly. Furthermore as per the previous design criteria, as little information as possible should be divulged to this consent authority, a protocol divulging only anonymous data would make it possible to employ a public blockchain as the consent authority.

3.4.2 Protocol

Note here that operations as arguments constitute a hash of one such operation.

`CollectionRequest(category, controller signature)`
A request to collect the data of the provided category.

`ProcessingRequest(CollectionRequest, method, controller signature)`
A request to process the category of data described in the `CollectionRequest` operation of the arguments, using the provided `method`.

`Consent(CollectionRequest | ProcessingRequest | Object, subject signature)`
The consent of the subject to either a collection request, processing request or a change of mind of an earlier objection.

`Object(Consent, subject signature)`
An objection to an earlier consent.

`Acknowledgment(Consent | Object, controller | registrant signature)`
An acknowledgment of a given consent or objection, this can either be signed by the controller or a trusted third party registrant.

Note here that `CollectionRequest` is the only operation that does not have a hash pointer in its arguments, which in this case means that, any operation will either be `CollectionRequest` or have one as their ancestor. An example of how this protocol might unfold can be seen in Figure 1.

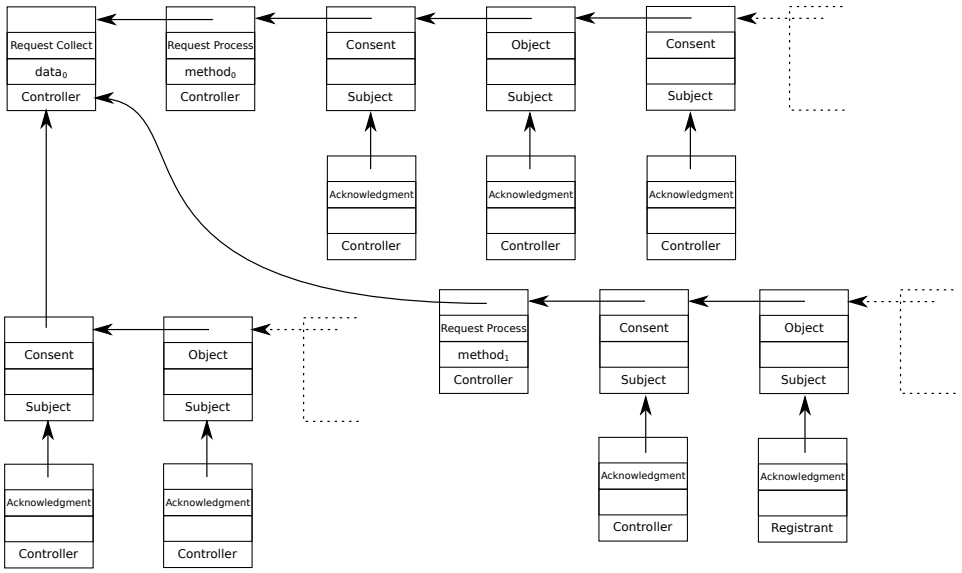

Figure 1: An example of how the block structure of the protocol might progress.

Data Subject Rights
Here are highlighted the rights of the data subject as described in the GDPR, which can be upheld by employing the protocol. Also shown is, how these rights can be upheld through the operations of the protocol.

Right of Access

The data subject shall have the right to obtain from the controller confirmation as to whether or not personal data concerning him or her are being processed, and, where that is the case, access to the personal data and the following information:

(a) The purpose of processing
When issuing a `ProcessingRequest` operation, the data field of the block can describe the purpose of processing.

(b) The categories of personal data concerned
When issuing a `CollectionRequest` the data field of the block can describe the categories of personal data concerned.

(d) The envisaged period for which the personal data will be stored
This can again be stored in the data field of the `CollectionRequest`.

(e) The existence of the right to request from the controller rectification or erasure of personal data or restriction of processing of personal data concerning the data subject or to object to such processing
Implicitly supported by employing this protocol.

(g) Where the personal data are not collected from the data subject, any available information as to their source
This can again be stored in the data field of the `CollectionRequest`.

Right to Object

The data subject can, at any point, object to a previous consent given. Whether the request is for collection or processing.

Right of Erasure

An objection to collection of a category of data is in the protocol equivalent to a request for erasure, as this objects to any collection of this category of data.

Right of Rectification

By objecting to the collection of the data that needs to be rectified, a collection request can then be sent for the rectified data. This would require the removal of erroneous data, the subject could then provide the rectified data.

Deriving Consent from the Consent Chain

We define R as a request, and R_d as a request of d, O is defined as an objection operation, C is defined as a consent operation, and lastly A is defined as an acknowledgment operation.

The consent chain, which in this case is a tree with the root being a collection request and nodes being operations, is defined as B. Two nodes $\{v, w\} \in B$ have the following relations and set operations

$v < w$	if v is a descendant of w
$v > w$	if v is an ancestor of w
$v \prec w$	if v is the child of w
$v \succ w$	if v is the parent of w
$<_v$	the set of all ancestors of v
$>_v$	the set of all descendants of v
$\leq_v$	the set v and all of its ancestors
$\geq_v$	the set v and all of its descendants

For a subtree of consents, objections and acknowledgments, we can define the consent as

$$\begin{aligned} SubConsent(T) = \neg \exists_{v \in T} \big[v = R\big] \wedge \\ \exists_{a,c \in T} \Big[c = C \wedge a = A \wedge a \prec c \wedge \\ \neg \exists_{v,w \in >_c} \big[v = O \wedge w = A \wedge w \prec v\big]\Big] \end{aligned}$$

Which in other words means that there should exist no other requests in the tree, and there should exist an acknowledged consent with no acknowledged objections in its descendants.

For a processing method or collection category d, and more specifically the request thereof R_d, consent can be defined as:

$$Consent(R_d) = \exists_{r,w \in T} \big[r = R_d \wedge w \prec r \wedge w = C \wedge SubConsent(\leq_w)\big]$$

Using this we can define the consent of a category and method pair c, m as:

$$Consent(c, m, B) = \exists_{v,w \in B} \big[v = R_c \wedge w = R_m \wedge v \succ w \wedge Consent(v) \wedge Consent(w)\big]$$

Dispute Resolution To resolve disputes between a subject and controller, on the matter of consent of a processing method on a category of data, the two parties can each produce a consent chain. The winner of the dispute will be the party which can produce the longest acknowledged consent chain.

3.4.3 Case Conclusion

To conclude on the design of this protocol, the design criteria will here be revisited and shown how they are achieved.

Indisputable Consent States

As described in the design criteria, it should be impossible to construct an indisputable consent state that the other party has not agreed to.

Forkability

One way to produce to produce two differing states of a consent relationship, would be to fork a consent chain. A chain is forked in the case where two different operations point to the same previous operation, which would result in two different consent chains depending on which tail of operations you chose to follow.

A way to ensure that these chains cannot be forked, is by making sure appending an operation to the chain is deterministic, *i.e.*, only one unique operation can be appended to any one consent chain.

Consent Chains

For a given request operation, there can exists a chain of consents and objections. This chain, as described in the protocol, is a linked list of alternating consents and objections. For this list to be unforkable, appending an element to it should be a deterministic process.

To prove this we look at the start of the chain. In the beginning there are no consent or objections on, the chain is of length 0, as defined in the previous section, this corresponds to a no consent state on the request.

As per the protocol, only a consent operation can be applied. The consent operation holds three pieces of information: the hash code of the request, a code corresponding to a consent operation, and a signature corresponding to the other two pieces of information. All of these pieces of information are unique. Constructing two different consent operations here are thus impossible and cannot lead to a fork.

If applying an operation to a chain, the head of the chain is either an object operation or a consent operation. Lets for example say that it is a consent operation that is the head, only an object operation can be applied. The object operation contains three pieces of information: the hash code to the head, a code corresponding to an object operation, and a signature corresponding to the other two pieces of information. Again all pieces of information are deterministic and cannot lead to a fork.

Applying a consent operation follows the same argument as above, and as such, is deterministic.

Acknowledgments are deterministic as well, but further applying operations to a consent chain does not depend on them. There can be two acknowledgements per operation on the consent chain, one by the controller and one from a consent authority, but each operation need only be acknowledged once, and

as discussed it changes nothing about the determinism of the next operation on the chain.

Requests
A request for collection of a category contains three pieces of information: the category of data to collect, the collection request code, and a signature of the controller. These are deterministic per category, thus it is impossible for the controller to create two different requests for the same category.

A request for processing using a method on a category of data contains four pieces of information: A hash code for the request of the category of data, a processing request code, the processing method, and a signature by the controller. Again are all values here deterministic for a category/method pair, thus it is impossible for the controller to create two different requests for the same category/method pair.

Two Party Consent Chain
As there is no way to fork the chain, there is no third party needed to reach consensus on the state of consents. The controller and subject can hold the chain themselves, and be sure that one party cannot change the state without informing the other party. In the event of dispute, the party that can produce the longest acknowledged chain wins.

Blockchain Consent Authority
If the consent authority is a smart contract on the Ethereum network[But13], this smart contract can be designed to log and acknowledge consent and objection operations. If the smart contract is agreed upon by the subject and the controller as the consent authority, the controller would then be obligated to consult the smart contract to check whether there are consent or objection operations logged on it that applies to one of their consent relationships.

The logged consent or objection operation contains only a hash code representing either a chain or a request, but it is impossible for anyone but the subject or the controller to know which. This means that even though the operation is public, the only information available is that someone either objected or consented to something, but not who consented or objected, and not what they consented or objected to.

This renders the logging of a consent or objection anonymous from the perspective of the smart contract. More specifically this anonymity hinges on whether or not the data subject can be connected to the entity logging the operation. In the case that a data subject can be connected to the logged operation, it is not possible for a third party to know what the logged operation relates to.

This shows that it is possible to devise a protocol for consent management which ensures appropriate rights of the data subject, and that it is in most cases sufficient, for only the subject and the controller to be involved. The data structure for this is inspired by blockchain design, and can be augmented using current blockchain technology to ensure reliability in the face of an unresponsive, reluctant or malicious data controller.

References

[Art14] Article 29 Data Protection Working Party. Opinion 05/2014 on anonymisation techniques, 2014.

https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf.

[Blo18] Blockchain and the GDPR. The european union blockchain observatory and forum, 2018. https://www.eublockchainforum.eu/sites/default/files/reports/20181016_report_gdpr.pdf.

[But13] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. https://github.com/ethereum/wiki/wiki/White-Paper, 2013.