Un modèle de tokenisation basé sur la chaîne de blocs pour la traçabilité et la conformité de la collecte de données

par

Alaeddine CHOUCHANE

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE
M. Sc. A.

MONTRÉAL, LE 13 JUILLET 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Kaiwen Zhang, directeur de mémoire
Département de génie logiciel logiciel et des TI, École de technologie supérieure

M. Chamseddine Talhi, co-directeur
Département de génie logiciel logiciel et des TI, École de technologie supérieure

M. Georges Kaddoum, président du jury
Département de génie électrique, École de technologie supérieure

M. Vincent Levesque, examinateur
Département de génie logiciel logiciel et des TI, École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 2 JUILLET 2020

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui ont contribué au succès de ma maîtrise et qui m'ont aidé lors de la rédaction de ce mémoire.

Je voudrais dans un premier temps remercier, mon directeur de mémoire M.Kaiwen Zhang, professeur au Département de génie logiciel et des technologies de l'information de l'École de technologie supérieure Montréal, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je remercie également mon co-directeur M. Chamseddine Talhi, professeur au Département de génie logiciel et des technologies de l'information de l'École de technologie supérieure Montréal, pour ses efforts, son encadrement efficace et son assistance tout au long mon projet de recherche.

Mes vifs remerciements s'adressent à mes amis et collègues du laboratoire "FUSÉE LAB" pour m'avoir soutenu durant ma maîtrise.

Enfin, je dédie ce mémoire à ma mère et mon père qui étaient à mes côté depuis mes débuts, mes frères et ma sœur pour leurs conseils ainsi que leur soutien moral, et à ma fiancée pour ses encouragements constants et pour les beaux moments qu'on a passé ensemble.

Un modèle de tokenisation basé sur la chaîne de blocs pour la traçabilité et la conformité de la collecte de données

Alaeddine CHOUCHANE

RÉSUMÉ

En tant que nouveau règlement sur la confidentialité, le règlement général européen sur la confidentialité des données (RGPD) perturbera l'industrie de la collecte de données qui doit respecter ses clauses en matière de transparence, de traçabilité et de gouvernance des données. En particulier, le RGPD confère trois droits importants aux individus : le droit à l'effacement, le droit à la portabilité des données et le droit à la restriction du traitement. Dans cet article, nous tirons parti de l'immuabilité offerte par les technologies de registres distribués (DLT), combinées à une exécution logique de validation transparente et fiable via des contrats intelligents, pour prendre en charge la collecte de données conforme au RGPD. Nous proposons un nouveau format de tokenisation des données, qui enregistre le consentement et fournit une piste d'audit capturant les flux de traitement des données. Nous montrons comment notre approche de tokenisation des données peut être combinée avec un système de stockage de fichiers décentralisé (comme IPFS) afin de stocker efficacement de gros volumes de données. Nous montrons également comment la suppression des données peut être prise en charge en manipulant le mécanisme de disponibilité des données fourni par IPFS. Nous décrivons une étude de cas pour la formation en apprentissage automatique, montrant la capacité de notre modèle à se conformer au RGPD. Nous implémentons notre modèle en utilisant Solidity, fournissons une évaluation des performances de notre système en utilisant Ethereum et IPFS, et réalisons une analyse de sensibilité des principales fonctions fournies par notre application décentralisée.

Mots-clés: DLT, confidentialité, RGPD, contrats intelligents

A Blockchain-Based Tokenization Model for Data Collection Traceability and Compliance

Alaeddine CHOUCHANE

ABSTRACT

As a new privacy regulation, the European General Data Privacy Regulation (GDPR) will disrupt the data collection industry which must comply with its clauses regarding transparency, traceability and data governance. In particular, the GDPR confers three important rights to individuals : the right to erasure, the right to data portability and the right to restrict processing. In this paper, we leverage the immutability offered by distributed ledger technologies (DLTs), combined with transparent and reliable validation logic execution via smart contracts, to support GDPR-compliant data collection. We propose a novel data tokenization format, which records consent and provide an audit trail capturing the data processing flows. We show how our data tokenization approach can be combined with a decentralized file storage system (such as IPFS) in order to efficiently store large volumes of data. We also demonstrate how data deletion can be supported by manipulating the data availability mechanism provided by IPFS. We describe a case study for machine learning training, showcasing the ability of our model to comply with GDPR. We implement our model using Solidity, provide an evaluation of our system performance using Ethereum and IPFS, and conduct a sensitivity analysis of the main functions provided by our decentralized application.

Keywords: DLT, privacy, GDPR, smart contracts

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

DLT	Destributed Ledger Technology
RGPD	Règlement Général sur la Protection des Données
IPFS	InterPlanetary File System
DHT	Destributed Hash Table
DAG	Directed Acyclic Graph
VANET	Vehicular Ad Hoc Network
TEE	Trusted Execution Environment
EVM	Ethereum Virtual Machine
ABI	Application Binary Interface
KAC	Key-Aggregate Cryptosystem
API	Application Programming Interface
CCPA	California Consumers Protection Act
LGPD	Lei Geral de Proteção de Dados (Loi générale sur la protection des données)

INTRODUCTION

La confidentialité est l'un des problèmes les plus difficiles liés à la collecte de données car il est notoirement difficile de détecter et de prévenir les fuites de données (Parno, McCune, Wendlandt, Andersen & Perrig, 2009). Les statistiques publiées par le Breach Level Index montrent que plus de 3 milliards d'enregistrements de données ont été compromis au cours du premier semestre 2018 [1]. Pour résoudre ce problème, le RGPD, un nouveau règlement sur la confidentialité des données, est entré en vigueur en mai 2018. La nouvelle loi européenne perturbe les industries traitant des données personnelles qui doivent se conformer au règlement ou s'exposer à des amendes excessives pouvant atteindre 4% du chiffre d'affaires mondial total de l'entreprise (Goddard, 2017).

De nos jours, les grandes entreprises sont de plus en plus axées sur les données. Le processus de collecte de données implique l'obtention de flux de données auprès de clients individuels (appelés propriétaires de données), la consommation des données pour obtenir des analyses et des informations, et le stockage des informations pertinentes dans un entrepôt de données. Les problèmes de conformité à la confidentialité des données obligent à fournir des systèmes de collecte de données qui permettent de vérifier la violation de la confidentialité conformément aux politiques de consentement données par les propriétaires de données. Les entreprises (processeurs de données) doivent conserver une piste d'audit du processus de traitement des données. Les processeurs de données doivent également répondre à la demande des propriétaires de données d'obtenir une copie de leurs données et d'effacer (oublier) leurs informations.

À la lumière de ces objectifs, nous considérons l'utilisation de la technologie des registres distribués (DLT) comme un bon candidat pour aider au respect de la confidentialité (Zhang & Jacobsen, 2018). Les DLT à base de chaîne de blocs sont bien adaptés pour l'audit des journaux de données en raison de l'immuabilité des chaînes de blocs fournie par des mécanismes cryptographiques.

[1] https://www.thalesesecurity.com/2019/data-threat-report

La conformité aux règles de confidentialité peut être soutenue par des contrats intelligents, qui sont des programmes auto-exécutables déclenchés à différentes étapes du processus de collecte de données. Enfin, la transparence fournie par les chaînes de blocs et les contrats intelligents permet à tous les propriétaires et processeurs de données de comprendre et d'approuver la logique de collecte des données.

Dans ce mémoire, nous proposons un nouveau modèle de tokenisation des données pour capturer une piste d'audit du processus de collecte de données et se conformer aux réglementations RGPD. Nous motivons la création de notre nouveau modèle car les normes de tokenisation établies, telles que ERC20 (Buterin & Vogelsteller, 2015) et ERC721 (Entriken, Shirley, Evans & Sachs, 2018), ne conviennent pas à des fins de collecte de données. Notre modèle de jeton de données capture le consentement de l'utilisateur, la relation entre les données source et la sortie résultante, ainsi que l'intégrité des données. Grâce à la chaîne de blocs, notre système s'adapte aux cas d'utilisations de *Big Data*.

Cependant, la chaîne de blocs elle-même est une solution incomplète pour deux raisons. Tout d'abord, l'immuabilité rend impossible pour les propriétaires de données de supprimer leurs données privées comme requis par la réglementation RGPD. Deuxièmement, les chaînes de blocs ne sont pas conçues pour stocker de gros volumes de données, comme ce serait le cas pour les applications Big Data. Par conséquent, nous proposons l'utilisation d'une couche de stockage secondaire hors chaîne pour fournir un entreposage de données efficace et effaçable. En particulier, nous utilisons le système de fichiers interplanétaire (IPFS) comme couche de stockage pour notre plateforme, qui est un réseau de stockage de fichiers pair à pair offrant une disponibilité réglable via la réplication, l'intégrité via l'adressage par hachage et des coûts d'hébergement moins chers que les solutions sur-chaîne. Nous montrons la manière d'intégrer IPFS dans notre solution de chaîne de blocs pour prendre en charge le droit d'effacement requis par le RGPD.

Dans ce mémoire, nous proposons une plateforme de collecte de données décentralisée conforme aux réglementations du RGPD concernant le consentement, la traçabilité du traitement des données et le droit d'effacement. Nos contributions sont les suivantes :

- nous proposons un nouveau modèle de tokenisation des données pour assurer la traçabilité et le consentement ;
- nous utilisons IPFS comme couche de stockage secondaire hors chaîne pour prendre en charge un stockage efficace des données et le droit d'effacement RGPD ;
- nous montrons comment la conformité au RGPD est obtenue avec notre solution et décrivons une étude de cas pour la formation en apprentissage automatique ;
- nous implémentons notre solution en tant que contrat intelligent en utilisant Solidity et évaluons ses performances en utilisant Ethereum.

Ainsi, nous avons organisé ce mémoire de la façon suivante. Nous présentons les concepts de base ainsi que les technologies relatives à notre projet dans le premier chapitre. Les travaux connexes sont regroupés et organisés sémantiquement dans le deuxième chapitre dans le but de positionner nos contributions par rapport aux solutions existantes. Dans le troisième chapitre, nous présentons notre modèle de tokenisation et l'architecture de notre système de collecte de données. Le chapitre 4 est consacré pour valider la conformité de notre solution avec le RGPD. L'évaluation des performances de notre système est détaillée dans le chapitre 5. Nous résumons notre expérience durant le parcours de recherche dans le chapitre 6.

CHAPITRE 1

NOTIONS DE BASE

Dans ce chapitre, nous passons d'abord en revue les principales technologies nécessaires à notre travail, notamment les chaînes de blocs, les contrats intelligents, Ethereum, les normes de tokenisation et IPFS. Ensuite, nous fournissons un résumé du RGPD et décrivons notre interprétation de la réglementation, qui est la base des exigences supportées par nos solutions. Notez que le RGPD et l'interprétation de ses règles peuvent changer au fil du temps ; nous laissons comme travaux futurs l'adaptation de notre solution actuelle à un modèle évolutif du RGPD.

1.1 Chaîne de blocs et les technologies de registre distribué

Les transactions et les contrats sont des éléments fondamentaux de nos systèmes économiques, juridiques et politiques. Pourtant, il s'avère que les systèmes actuels qui les enregistrent et les gèrent ne suivent pas l'évolution de l'économie numérique. En fait, ces systèmes ne répondent plus aux besoins des nouvelles grandes applications de données en termes de disponibilité, d'évolutivité et de traçabilité efficace des enregistrements. La chaîne de blocs se présente en tant que solution à ces problèmes en offrant un registre de données fiable et transparent. La chaîne de blocs permet de réduire le coût de déploiement des nouvelles solutions. En effet, elle remplace les infrastructures de sauvegarde actuelles par un système d'enregistrement basé sur un réseau pair à pair.

Se servant de la technologie sur laquelle se basent Bitcoin, Ethereum et d'autres crypto-monnaies, la chaîne de blocs est un registre distribué de données qui enregistre les transactions répliquées dans un réseau pair à pair décentralisé. Une fois dans le grand livre, les transactions sont permanentes, vérifiables et inviolables.

Comme illustré dans la figure 1.1, les transactions sont collectées ensemble dans des blocs qui sont liés cryptographiquement pour former la chaîne de blocs. L'entête du bloc contient des

méta-données incluant le hachage du block précédent. La nature distribuée de la chaîne de blocs élimine la nécessité d'une unité centrale comme les institutions financières et donc le besoin de confiance. La permanence et l'inviolabilité des blocs à l'intérieur de la chaîne de blocs sont assurées par le hachage, combiné à une validation par un protocole de consensus.

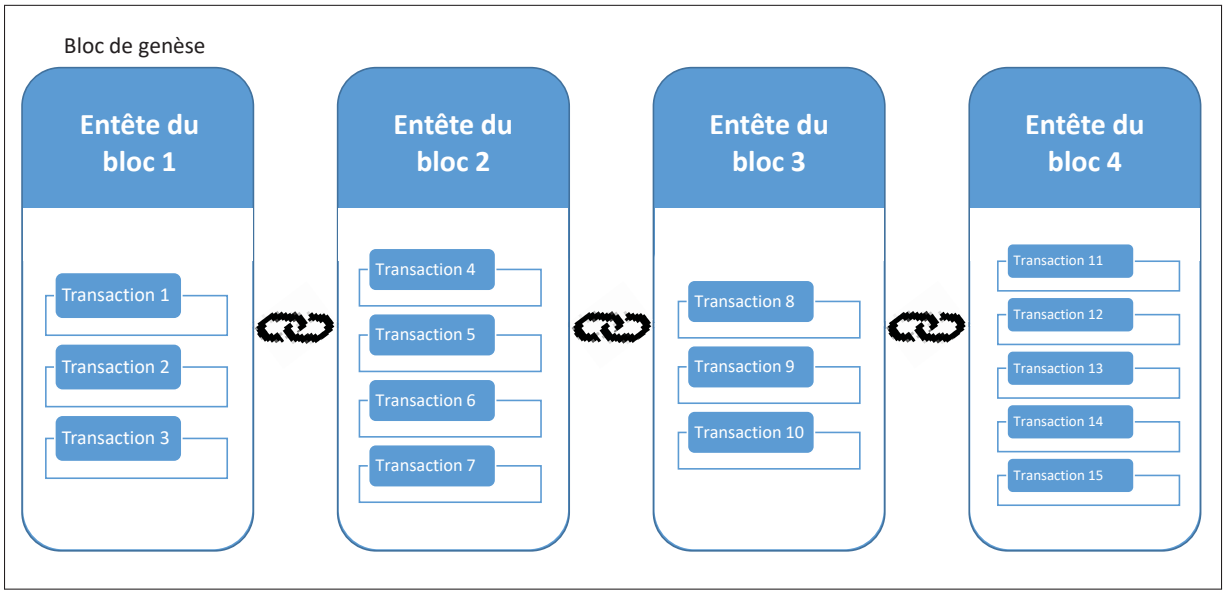

Figure 1.1 Structure de base de la chaîne de blocs

1.1.1 Bitcoin

Fondée en 2008 par Satoshi Nakamoto, Bitcoin est la plateforme de cryptomonnaie la plus célèbre. C'est un système pair à pair basé sur la chaîne de blocs qui permet un échange de la monnaie d'une façon décentralisée. Contrairement au système bancaire centralisé qui nécessite une institution financière pour effectuer une transaction digitale entre deux parties, Bitcoin assure un transfert direct de la monnaie entre les parties.

1.1.1.1 Minage

L'ajout des blocs à la chaîne de blocs passe par un processus qui s'appelle le minage (Nakamoto et al., 2008). Pour ce faire, les mineurs (nœuds participants à la chaîne de blocs) reçoivent les

transactions diffusées par les clients et les collectent dans des blocs. Ensuite, chaque nœud commence à résoudre la preuve de travail de son bloc. Une fois résolue, le mineur diffuse son bloc aux autres nœuds qui vérifient sa validité. Un bloc est considéré valide s'il a une preuve de travail valide et si toutes transactions contenues ne sont pas déjà consommées dans des blocs antérieurs. Les mineurs expriment leurs acceptations d'un bloc en commençant à travailler sur un nouveau bloc en se basant sur ce dernier.

1.1.1.2 Preuve de travail

Bitcoin est une plateforme publique, sans permission et sans unité de contrôle. Cela peut engendrer des failles de sécurité, notamment, le problème de double dépenses qui consiste à dépenser le même jeton de monnaie dans deux transactions différentes. Pour y remédier, Bitcoin utilise un mécanisme de consensus qui s'appelle preuve de travail (proof of work). Ce mécanisme repose sur la résolution d'un problème cryptographique qui consomme du temps pour le resoudre (10 minutes pour Bitcoin) mais qui est facile à vérifier. De ce fait, chaque bloc doit passer par la preuve de travail pour qu'il soit ajouté à la chaîne. Comme les nœuds honnêtes ne minent de nouveaux blocs que sur la plus longue chaîne connue, il est impossible pour un nœud malveillant d'ajouter des blocs avec des transactions altérées à moins de détenir 51% du réseau, ce qui est peu probable étant donné l'échelle mondiale du réseau Bitcoin actuel.

1.1.2 Ethereum et les contrats intelligents

Ethereum est conçu en 2014 pour mettre en place un environnement qui permet de créer des applications distribuées dépassant la cryptomonnaie tout en exploitant les avantages de la chaîne de blocs. Ethereum est un système informatique décentralisé qui permet d'exécuter des programmes appelés contrats intelligents. Il se base sur la chaîne de blocs pour synchroniser l'état du système entre tous les nœuds participants ainsi que pour utiliser sa cryptomonnaie appelée *ether* (Antonopoulos & Wood, 2018). Ethereum fournit une couche d'application décentralisée au-dessus de la chaîne de blocs pour prendre en charge les opérations au-delà de la cryptomonnaie (Cuccuru, 2017). Cela est possible grâce à des contrats intelligents autonomes

exécutés par des transactions (Buterin et al., 2013). Étant donné que le bytecode exécutable du contrat intelligent est lui-même stocké sous forme de données en chaîne, la propriété d'immuabilité des chaînes de blocs garantit une exécution fiable du programme de contrat intelligent. De plus, la transparence du bytecode exécutable permet de vérifier son intégrité par rapport au code source compilé. Il existe deux types de comptes dans Ethereum, chaque compte ayant une adresse de 20 octets :

- **compte détenu en externe :** Ce compte est contrôlé par un utilisateur Ethereum et possède une paire de clés publique/privée ;
- **compte de contrat :** Ce compte appartient à un contrat intelligent et est contrôlé par son code.

Un compte de contrat contient une combinaison de bytecode de contrat intelligent et de données spécifiques au contrat et est déployé par un compte externe sur la chaîne de blocs. Les contrats intelligents sont écrits en code Ethereum Virtual Machine (code EVM) et fournissent une interface binaire d'application (ABI) qui permet aux utilisateurs d'interagir avec elle. Un appel de fonction de contrat inclus dans une transaction est exécuté par le mineur qui a inclus la transaction dans un bloc, et est vérifié par chaque nœud qui obtient une copie du bloc.

Afin d'envoyer des pièces à un autre compte, de déployer un contrat intelligent ou d'exécuter le contrat, un compte externe crée une transaction qui contient les champs indiqués dans la tableau 1.1.

Le gaz est l'unité permettant de calculer le coût de chaque opération à Ethereum. Cela comprend l'exécution des codes d'opération (Opcodes) dans une fonction de contrat intelligent et le stockage des données dans l'état du compte. Le prix du gaz est calculé en ether et est librement défini par l'expéditeur de la transaction ; cependant, les utilisateurs devraient définir un prix élevé de gaz afin d'encourager les mineurs à ajouter rapidement leurs transactions à un bloc.

Tableau 1.1 Structure d'un compte externe

Champ	Description
nonce	Le nombre de transactions envoyées par le compte. Il incrémente chaque fois que ce compte envoie une transaction
prix du gaz	Le prix d'une unité de gaz que l'utilisateur a l'intention de payer
limite de gaz	Le nombre maximum d'unités de gaz que l'utilisateur autorise à utiliser dans cette transaction
adresse de destination	Nombre de publications reçus sur MQTT
valeur	La quantité d'éther que l'utilisateur veut envoyer
signature	La signature cryptographique de l'expéditeur de la transaction

1.1.3 Hyperledger

Contrairement à Bitcoin et Ethereum, Hyperledger est une solution open source pour des chaînes de blocs avec permission, à savoir une chaîne de blocs qui demande une autorisation d'une entité de contrôle pour permettre à un nœud d'entrer au réseau. Hyperledger fournit un ensemble de technologies de chaîne de blocs adaptés au contexte d'entreprise telles que les registres distribués, les moteurs de contrats intelligents et les interfaces applicatives. Il existe plusieurs implémentations de Hyperledger telles que Hyperledger Fabric, Hyperledger Indy et Hyperledger Iroha. La majeure différence entre ces implémentations est le mécanisme de consensus utilisé.

1.1.4 Comparaison des systèmes et discussion

1.1.4.1 Ethereum et Bitcoin

Ether et Bitcoin, les cryptomonnaies de Ethereum et Bitcoin, sont les jetons digitaux les plus populaires sur le marché des cryptomonnaies. Les deux monnaies ont plusieurs similarités dont la nature décentralisée vue qu'elles ne sont pas gérées par une unité centrale. Aussi, Les deux monnaies se basent sur la chaîne de blocs pour la distribution et les transactions. En termes de structure du réseau, Bitcoin et Ethereum se basent sur une chaîne de blocs publique, et utilisent le même mécanisme de consensus qui est la preuve de travail.

Cependant, les deux systèmes sont techniquement différents à bien des égards. Les transactions dans Ethereum peuvent contenir du code à exécuter, tandis que les transactions de Bitcoin contiennent de scripts basiques qui indiquent la façon d'accéder aux Bitcoins. De plus, un bloc prend entre 10 et 12 secondes pour être ajouté à la chaîne de Ethereum, alors que cela prend 10 min pour qu'il soit ajouté à la chaîne de Bitcoin. Mais surtout, Bitcoin et Ethereum diffèrent en ce qui concerne leurs objectifs. Bitcoin a été créé comme un système monétaire alternatif au système centralisé existant, tandis que Ethereum a été conçu pour compléter Bitcoin en facilitant la création des applications décentralisées monétisées par la cryptomonnaie.

1.1.4.2 Ethereum et Hyperledger

Ethereum et Hyperledger sont basés sur la technologie de chaîne de blocs. Ils utilisent de protocoles de consensus pour maintenir la chaîne à jour et pour garantir l'immutabilité de la chaîne. Les deux systèmes sont conçus pour créer des applications décentralisées à base de contrat intelligents. Une des différences majeures entre Ethereum et Hyperledger est le type de chaîne de blocs utilisé. Ethereum est basé sur une chaîne de blocs publique ce qui le rend plus adapté au applications B2C (Business to Consumer) puisque toute personne peut participer aux transactions. Au contraire, Hyperledger utilise une chaîne de blocs avec permission ce qui le rend plus adapté au contexte B2B (Business to Business) puisque la participation demande une autorisation. D'autre part, Ethereum fournit sa cryptmonnaie et utilise la preuve de travail pour le consensus, tandis que Hyperledger n'a pas de cryptomonnaie et utilise différents mécanismes de consensus dépendamment de l'implémentation.

1.1.4.3 Discussion

Dans notre travail, nous utilisons les contrats intelligents pour coder notre modèle de collecte de données et nous utilisons le coût du gaz comme mesure pour évaluer les performances du système et fournir une analyse de sensibilité de divers paramètres afin de comprendre le modèle d'utilisation optimal de notre solution, bien que notre travail est actuellement implémenté à l'aide de Solidity et déployé sur Ethereum, le modèle proposé est général et peut être implémenté avec

d'autres langages. Nous avons choisi Solidity car il est actuellement le langage de programmation de contrat intelligent le plus mature et bénéficie d'une solide communauté de développement et de kits d'outils. Par conséquent, Ethereum est un choix naturel puisqu'il prend en charge Solidity nativement. Cependant, selon l'application de collecte de données spécifique, un système de chaîne de blocs différent peut être souhaitable, tel qu'un système autorisé (Zhang & Jacobsen, 2018). Dans ce cas, nous pouvons ré-implémenter ou adapter notre code Solidity pour travailler sur une plateforme différente. Par exemple, Quorum prend en charge le langage Solidity dans une configuration autorisée (Baliga, Subhod, Kamat & Chatterjee, 2018).

1.2 InterPlanetary File System

IPFS (InterPlanetary File System) a été initialement publié en 2015 avec l'intention de construire une infrastructure de système de fichiers décentralisée pour le futur Web. IPFS combine des éléments de systèmes distribués ayant connu du succès, à savoir les DHT, BitTorrent et Git (Benet, 2014).

Étant donné qu'IPFS est basé sur un réseau pair à pair public et peu fiable, l'adressage de données basé sur la localisation traditionnel peut entraîner deux problèmes :

- un problème d'intégrité : La validité du contenu d'un fichier n'est pas vérifiée ;
- un problème de disponibilité : Un contenu n'est plus accessible si sa localisation n'est plus disponible.

Pour surmonter ce problème, IPFS utilise un adressage basé sur le contenu, qui consiste à calculer un hachage du fichier qui sera utilisé pour la recherche. De cette façon, si nous voulons vérifier l'intégrité d'un fichier, nous n'avons qu'à recalculer son hachage et le comparer avec le hachage utilisé comme adresse.

Lorsqu'un fichier est ajouté à IPFS, il est stocké dans des blocs sur le répertoire IPFS d'un nœud et ces blocs sont liés ensemble dans un graphique acyclique dirigé (DAG). Pour localiser un fichier adressé au contenu, IPFS utilise la *Kademlia* table de hachage distribuée (DHT) (May-

mounkov & Mazieres, 2002). Ce mécanisme peut trouver le peerID qui fournit un certain bloc ou un fichier. Pour optimiser l'espace de stockage, chaque nœud IPFS possède un ramasse-miettes qui supprime périodiquement tous les blocs non épinglés. Ainsi, si nous voulons garantir la persistance d'un fichier, nous devons explicitement l'épingler. Un fichier peut ensuite être non épinglé afin d'être supprimé ou récupéré. Des services d'épinglage existent pour garantir la disponibilité d'un certain fichier en échange d'un coût périodique.

IPFS est utilisé dans notre système pour stocker des données en raison des propriétés d'intégrité fournies par son schéma de hachage de contenu. En outre, il est utilisé comme une alternative de stockage de fichiers moins chère en raison du coût élevé du stockage persistant sur la chaîne de blocs Ethereum. Enfin, nous pouvons tirer parti du mécanisme épinglage pour contrôler la disponibilité des données pour la conformité au RGPD.

1.3 Règlement général sur la protection des données

Le RGPD (Règlement Général sur la Protection des Données) est un nouvel ensemble de réglementations pour la protection de la confidentialité des données qui est entré en vigueur le 25 mai 2018. Le RGPD est introduit en réaction aux entreprises collectant une quantité massive d'informations sensibles auprès des citoyens européens sans explication claire sur la façon dont ces données sont traitées.

1.3.1 Les acteurs principaux

La figure 1.2 illustre les entités mentionnées dans RGPD. Les acteurs dans RGPD sont les éléments qui participent au processus de collecte de données que ce soit en générant, collectant, traitant ou supervisant le déroulement d'assemblage de données privées. En se basant sur le règlement (UE) 2016/679 du parlement européen et du conseil (Regulation, 2016), le tableau 1.2 présente les acteurs qui nous intéressent dans notre travail.

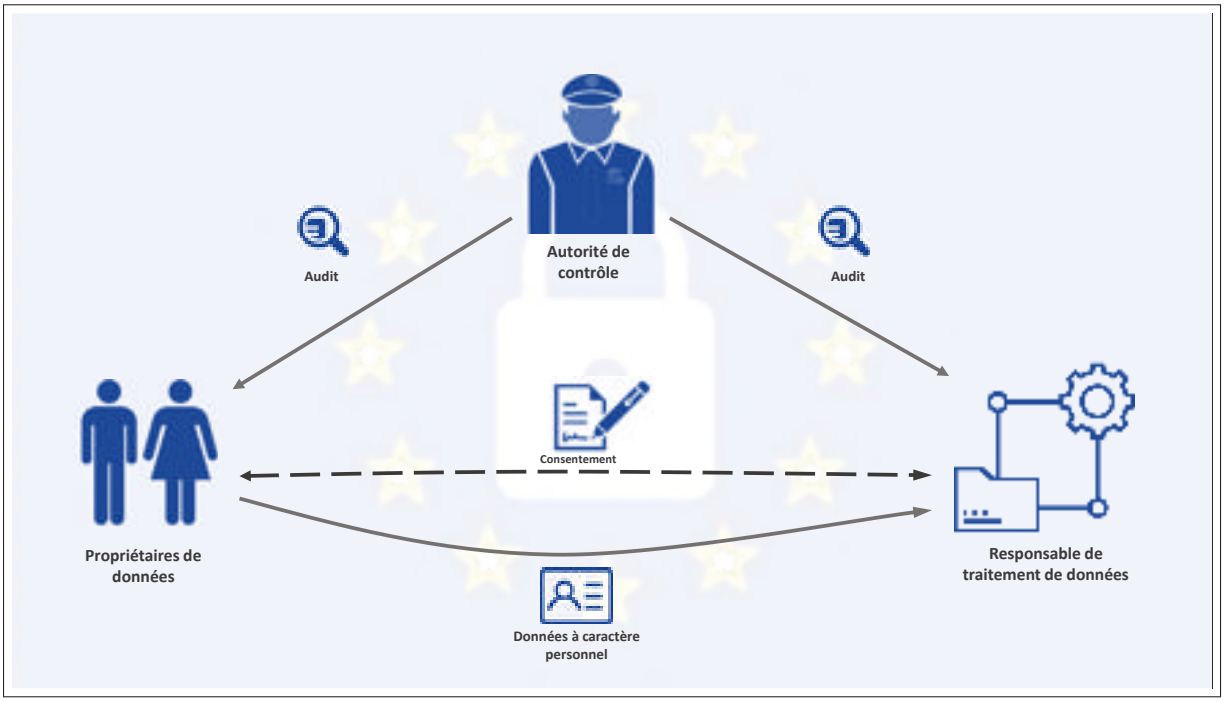

Figure 1.2 Les acteurs de RGPD et leurs interactions

Tableau 1.2 Les acteurs principaux de RGPD

Champ	**Description**
responsable de traitement	Selon l'article 4(7), le responsable de traitement et l'entité qui détermine l'objectif et la façon de traiter les données personnelles
données à caractère personnel	Selon l'article 4(1), les données à caractère personnel sont les données qui peuvent identifier une être physique identifiable dénommée dans notre travail "propriétaire de données"
autorité de contrôle	Selon l'article 51, l'autorité de contrôle est l'entité désignée par un État membre pour surveiller le respect du RGPD

1.3.2 Les pénalités infligées

Avec la nouvelle réglementation, les entreprises doivent augmenter le niveau de transparence de leurs processus de collecte de données. Les responsables de traitement de données seront

régulièrement audités pour la conformité au RGPD. Des sanctions peuvent être imposées selon la nature, la gravité et la durée de l'infraction. En particulier, une brèche à un consentement donné par un propriétaire de données mène à des sanctions pouvant atteindre jusqu'à 4 % du chiffre d'affaires mondial annuel. Une conséquence importante du RGPD est son applicabilité extraterritoriale ((Goddard, 2017)), ce qui signifie que les règles du RGPD s'appliquent aux organisations qui traitent des données européennes indépendamment de l'emplacement géographique des processeurs de données.

1.3.3 Les droits des propriétaires de données

Selon le même règlement cité en haut, les particuliers (propriétaires de données) bénéficient des droits suivants des entreprises (responsables de traitement) :

- le droit à la restriction et au contrôle d'accès aux données : les propriétaires de données ont le droit de contrôler l'accès aux données en accordant un consentement clair et direct au responsable du traitement des données. En outre, les propriétaires de données ont le droit de limiter et de restreindre l'accès accordé à un processeur de données ;
- le droit à l'effacement : les propriétaires de données ont le droit de supprimer leurs données personnelles afin qu'elles ne soient plus accessibles au responsable du traitement. La demande doit être traitée dans le mois suivant sa réception et sans retards injustifiés ;
- le droit au transfert : les propriétaires de données ont le droit de recevoir leurs données dans un format lisible par machine et ont le droit de les envoyer à un autre processeur de données.

Il est à noter que le traitement des données confidentielles n'est considéré légal que si le propriétaire de données a donné son consentement au traitement pour un ou plusieurs objectifs spécifiques. De plus, le propriétaire de données a le droit d'annuler son consentement à tout moment.

1.4 Normes de tokenisation

Il existe déjà de nombreuses normes qui permettent aux développeurs de contrats intelligents de créer des jetons. Ces normes fournissent des fonctions obligatoires à mettre en œuvre afin de créer des jetons interopérables. Parmi ces normes, nous trouvons ERC20 qui est une directive pour la création de jetons *fongibles* (Buterin & Vogelsteller, 2015). Ces jetons peuvent prendre la forme de crypto-monnaies, d'actions ou de toute autre ressource dénombrable interchangeable et divisible. Ce modèle ne convient pas à la collecte de données en raison de ses propriétés de non-unicité et de divisibilité qui font que deux jetons de même valeur valent la même chose (un billet de $10 vaut la même chose qu'un autre billet de $10). Dans notre contexte, nous souhaitons capturer des données d'utilisateurs individuels à l'intérieur de jetons qui sont identifiables de manière unique, et donc non fongibles.

ERC721 est une autre norme qui prend en charge les jetons non fongibles (Entriken *et al.*, 2018). Ces jetons représentent des actifs qui sont uniques, non interchangeables et non divisibles comme des objets de collection, des certificats de diamant ou des documents d'identité. Depuis sa sortie en septembre 2017, de nombreuses applications de jeux basées sur la chaîne de blocs ont été développées comme *CryptoKitties* et *EtherTulips*. Bien que l'ERC721 satisfasse aux exigences non fongibles de la collecte de données, il possède d'autres fonctionnalités qui ne sont pas compatibles, que nous étudierons dans la section suivante.

1.5 Conclusion

Dans ce chapitre, nous avons étudié certaines notions de base liées à notre travail. D'abord, nous avons présenté les plateformes de chaîne de blocs les plus utilisées (Badr, Horrocks & Wu, 2018) et nous avons discuté leurs points de similitude et de différence. Ensuite, nous avons introduit les fonctionnalité d'IPFS en tant que système de fichiers distribué. Puis, nous avons présenté notre interprétation des règlements de RGPD. Finalement, nous avons étudié deux normes existantes de développement de jetons avec des contrats intelligents.

CHAPITRE 2

REVUE DE LITTÉRATURE

Avec l'émergence de la chaîne de blocs et l'introduction des contrats intelligents, plusieurs travaux de recherche étudient des applications potentielles de cette technologie dans différents domaines. D'autre part, avec l'entrée en vigueur de RGPD, plusieurs recherches se concentrent sur les droits assurés par le règlement et les techniques à utiliser pour s'y conformer.

Dans ce chapitre, nous passons en revue les principales catégories d'œuvres liées à notre travail de recherche. Tout d'abord, nous examinons les documents qui exploitent un contrat intelligent pour préserver la confidentialité et le contrôle d'accès. Ensuite, nous examinons des articles axés sur la collecte de données. Finalement, nous présentons certains travaux qui examinent la compatibilité de la chaîne de blocs avec le RGPD.

2.1 Confidentialité et contrats intelligents

La technologie chaîne de blocs est utilisée dans de nombreux articles de recherche sur la confidentialité des données et le contrôle d'accès. FairAccess (Ouaddah, Abou Elkalam & Ait Ouahman, 2016) est un système de contrôle d'accès qui définit un jeton d'autorisation comme une devise qui représente le droit d'accès accordé par le créateur de la transaction à son destinataire. La solution est basé sur Bitcoin et assure principalement deux fonctionnalités importantes : Le partage direct des données personnelles, et le contrôle d'accès complet des données par le propriétaire. CreditCoin (Li, Liu, Cheng, Qiu, Wang, Zhang & Zhang, 2018a) est un protocole d'annonce incitatif basé sur la chaîne de blocs utilisé pour les VANET (véhicules ad-hoc networks). Il s'appuie sur la crypto-monnaie pour encourager les utilisateurs à faire des annonces de trafic. La confidentialité des annonceurs est protégée par un mécanisme de signature d'anneau de seuil. Cependant, l'utilisation d'un gestionnaire de trace pour identifier les utilisateurs malicieux suspects constitue une atteinte à la confidentialité des utilisateurs. ShadowEth (Yuan, Xia, Chen, Zang & Xie, 2018) et Hawk (Kosba, Miller, Shi, Wen & Papamanthou, 2016) présentent un système de contrat intelligent préservant la confidentialité qui s'appuie sur le TEE (Trusted

Execution Environment) pour gérer les données privées. La majeure différence entre les deux approches est que (Kosba *et al.*, 2016) divise le programme Hawk en une partie privée, exécutée dans le TEE, et une partie publique ; tandis que (Yuan *et al.*, 2018) déploie le contrat dans l'environnement de confiance et conserve l'identification du contrat dans la chaîne de blocs publique. (Wu, Williams & Perouli, 2019) explique comment la chaîne de blocs peut être une solution alternative pour la confidentialité dans le domaine de la santé. Les auteurs soutiennent que la chaîne de blocs peut être utilisée pour le contrôle d'accès aux dossiers de santé. Contrairement à ces travaux précédents, notre recherche se concentre sur le respect de la vie privée du point de vue du RGPD, se concentre sur les droits du consentement, du transfert et de l'effacement, et met en valeur la traçabilité de la collecte de données.

Dans (Wu *et al.*, 2019), les auteurs construisent un registre public basé sur la chaîne de blocs pour la vérification de la conformité aux politiques d'utilisation des données privées. La vérification de la conformité à la politique de confidentialité est effectuée sur un canal hors-chaîne appelé VOCMC (Verifiable Off-Chain Message Channel) qui implique deux ou plusieurs parties qui doivent se mettre d'accord sur la conformité ou la violation de la politique en fonction de leur propres versions des preuves de conformité à la politique. Le résultat de la négociation de l'accord est ensuite signé et stocké sur la chaîne de blocs. Ce processus crée un dossier public de conformité à la confidentialité qui permet aux organisations de se bâtir une réputation de confiance lorsqu'elles respectent leurs politiques de confidentialité. Les auteurs proposent un schéma pour vérifier la conformité à la politique de confidentialité en utilisant un modèle abstrait des réglementations RGPD. Dans notre travail, nous nous concentrons sur les détails de la réglementation RGPD concernant les droits des propriétaires de données et nous construisons un système de collecte de données conforme à notre interprétation de ces droits.

2.2 Collecte et partage de données

De nombreuses recherches ont abordé le problème de la collecte et du partage des données, en particulier dans le cloud. Chu et. Al. ont proposé dans (Chu, Chow, Tzeng, Zhou & Deng, 2013) un système de partage de données efficace dans le stockage cloud en créant un cryptosystème

à agrégats de clés (KAC) qui permet de partager des fichiers chiffrés qui sont stockés dans le cloud avec contrôle d'accès en fournissant une seule clé d'agrégation de déchiffrement. La taille de cette clé est constante et indépendante du nombre de fichiers à partager.

Dans (Sundareswaran, Squicciarini & Lin, 2012), les auteurs ont abordé le manque de contrôle du propriétaire des données dans les services cloud en créant une structure de responsabilisation des informations pour garder une trace de l'utilisation réelle des données des utilisateurs. La solution applique l'autorisation d'accès aux données en fonction de politiques définies par l'utilisateur et de deux modes de journalisation pour l'audit et le suivi de l'historique d'utilisation. Mona (Liu, Zhang, Wang & Yan, 2012) est un schéma qui permet le partage de données multipropriétaire entre des groupes dynamiques dans le cloud. Ce schéma permet l'ajout et la révocation d'utilisateurs, la création de données, la suppression et l'accès aux données dans un groupe de partage en tenant compte d'une liste de révocation. Dans notre travail, nous proposons une approche alternative de la traçabilité et du contrôle d'accès à l'aide de chaîne de blocs qui n'est pas nécessairement spécifique à l'utilisation du cloud.

De plus, de nombreux articles existent sur la collecte et le partage de données assistées par chaîne de blocs. ProvChain (Liang, Shetty, Tosh, Kamhoua, Kwiat & Njilla, 2017) est une architecture qui permet la collecte et la validation de la provenance des données (c'est-à-dire l'historique de création et d'opération exécuté sur un produit de données cloud). Les enregistrements de provenance des données sont collectés et stockés dans la chaîne de blocs pour garantir leur intégrité. Dans (Li, Barenji & Huang, 2018b), les auteurs ont créé un système de partage de données centré sur l'utilisateur pour les applications mobiles de soins de santé. Les prestataires de soins de santé et la compagnie d'assurance demandent au propriétaire l'accès au dossier de santé. Les données de santé et médicales sont hachées et stockées dans la chaîne de blocs pour assurer l'intégrité. Les demandes d'accès et les activités sont également enregistrées dans la chaîne de blocs à des fins d'audit.

Contrairement à ces travaux précédents, notre travail s'appuie sur des contrats intelligents pour définir un modèle de collecte de données et intègre la chaîne de blocs et IPFS ensemble pour prendre en charge les principaux droits du RGPD.

2.3 Chaîne de blocs et RGPD

Avec l'introduction de RGPD, plusieurs travaux de recherche se sont concentrés sur les corrélations potentielles entre le règlement européen et la chaîne de blocs. D'autres œuvres ont étudié des systèmes basés sur la chaîne de blocs qui se conforment au RGPD.

Les auteurs de (Ramsay, 2018) ont examiné l'applicabilité de RGPD dans un contexte de chaîne de blocs publique. À cette fin, ils ont énuméré les données personnelles dans la chaîne de blocs publique (clés publiques, le contenu des transactions et les adresses des nœuds). Ensuite, ils ont étudié la conformité de la chaîne de blocs avec les lois de RGPD. La recherche a abouti au fait que la chaîne de blocs est bien conforme avec les principes de protection des données, intégrité et transparence. Cependant, l'immutabilité de la chaîne de blocs contredit le droit à la suppression des données personnelles. Nous partageons la même interprétation que cette étude par rapport à la compatibilité entre la chaîne de blocs la RGPD. Pour assurer le droit à la suppression, nous utilisons un système de stockage hors-chaîne.

(van Geelkerken & Konings, 2017) explique comment la chaîne de blocs peut renforcer le respect des droits des propriétaires des données tels que cités dans la RGPD. L'étude divisent les chaînes de blocs en des registres transparents (publics) et opaques (privés) tout en discutant la faisabilité d'appliquer les droits de modification et d'effacement. Pour le registre transparent, on peut assurer le droit de modification de donnée en ajoutant les informations mises à jour dans des nouveaux blocs. Pour le droit d'effacement, on peut mettre des données encryptées dans la chaîne de blocs et supprimer la clé privée lorsqu'on reçoit une demande de suppression de données. Quant au registre opaque, l'assurance des droits est plus facile vu que les chaînes de blocs privées sont gérées par des autorités de régulation. De notre point de vue, les solutions proposées n'assurent pas vraiment les droits de modification et l'effacement vu que, pour le

premier, la mise à jour des données dans des nouveaux blocs n'élimine pas l'accès et l'utilisation des données des anciens blocs, et pour le deuxième, il faut trouver un moyen pour prouver la suppression de la clé de décryptage.

Les auteurs de (Truong, Sun, Lee & Guo, 2019) mettent en place un mécanisme de gestion de données conforme au RGPD pour les fournisseurs de service et les propriétaires de données. La solution est basée sur Hyperledger Fabric qui une chaîne de blocs avec permission. Un jeton d'accès est utilisé pour assurer l'accès aux données personnelles et un système de stockage de confiance est utilisé pour l'enregistrement des données. La solution proposée est entourée de certaines limitations dont l'utilisation de Hyperledger Fabric qui n'est pas publique et qui utilise un mécanisme de consensus exigeant que la majorité des nœuds participant soit honnête. Contrairement à ce travail de recherche, notre système est basé sur Ethereum et nous utilisons un système de stockage décentralisé (IPFS).

2.4 Conclusion

Dans ce chapitre non avons passé en revue des articles et des œuvres qui ont adressé des problèmes pertinents à notre travail de recherche. Nous avons étudié des solutions pour la confidentialité basées sur la chaîne de blocs, des systèmes de collecte de données et des recherches qui portent sur le RGPD et chaîne de blocs. D'après les travaux connexes, il est clair que la chaîne de blocs est une solution envisageable pour faire face au problème de confidentialité dans la collecte de données personnelles. Nous avons constaté aussi, que jusqu'au moment de la rédaction du mémoire, le nombre de systèmes conformes au RGPD et basés sur la chaîne de blocs est limité. Cela fait de notre travail l'un des premiers qui adresse ce sujet.

CHAPITRE 3

SOLUTION ET ARCHITECTURE PROPOSÉES

Dans ce chapitre, nous analysons d'abord ERC721 dans le contexte de la collecte de données. Nous présentons ensuite notre nouveau modèle de tokenisation de données et une architecture de référence déployée sur Ethereum.

3.1 Analyse de la norme ERC721

Bien qu'elle fournisse des jetons non fongibles, la norme ERC721 est incompatible avec nos exigences fonctionnelles, car les fonctions de l'API sont conçues pour des jetons qu'on peut collecter et échanger, ce qui n'est pas le cas pour le besoin de collecte de données. Dans notre cas, le processeur de données cherche à obtenir le consentement des propriétaires de données pour accéder et utiliser le contenu de leurs jetons de données, mais pas pour prendre la possession de ces jetons.

La figure 3.1 montre l'interface d'un jeton ERC721. Nous nous concentrons sur les opérations incompatibles suivantes indiquées dans le tableau 3.1.

3.2 Modèle de tokenisation des données proposé

L'objectif principal de notre solution est d'établir un système de collecte de données qui assure les droits des propriétaires de données conformément au RGPD tout en gardant une trace immuable de la séquence d'opérations. Afin de surmonter l'incompatibilité de la norme ERC721, nous proposons un nouveau modèle de jeton qui permet aux acteurs participants de créer des jetons contenant de nouvelles données, d'accorder le consentement à d'autres acteurs pour traiter les données d'un jeton et de lier les jetons de données ensemble pour représenter les flux de traitement des données. Comme le montre la figure 3.2, notre modèle de jeton présente deux structures de données : un jeton de données, et un consentement.

Tableau 3.1 Description des opérations incompatibles de ERC721

Opération	**Description**
balanceOf	Cette fonction renvoie le nombre de jetons collectés par un certain compte. Nous n'avons pas besoin de cette fonction dans notre modèle car nos jetons ne représentent pas un actif à collecter par les utilisateurs.
safeTransferFrom transferFrom	Ces fonctions obligatoires permettent le transfert de propriété des jetons d'un compte à un autre. Dans notre travail, nous n'exigeons pas le transfert de propriété des données personnelles. Au lieu de cela, l'utilisateur devrait pouvoir accorder l'accès à son jeton à une autre entité (le processeur de données).
approve setApprovalForAll getApproved isApprovedForAll	Ces fonctions sont utilisées pour permettre à une autre entité de gérer un jeton au nom d'un autre utilisateur. Pour la collecte de données, un propriétaire de données n'a pas besoin que ses jetons soient contrôlés par une autre partie. Le droit de consentement permet au processeur de données de créer ses propres jetons de données et de lier ensuite les jetons nouvellement créés aux jetons du propriétaire des données. Cependant, il ne lui permet pas de manipuler directement ces jetons de données.

Un jeton représente les données confidentielles. Il contient des méta-données qui lient le jeton aux données personnelles et aux jetons d'input. Ainsi, le jeton proposé contient principalement :

- un identifiant (ID),
- un hachage du contenu des données utilisé comme adresse IPFS,
- une liste des ID des jetons d'input.

En outre, un jeton contient deux variables booléennes pour représenter sa disponibilité et son état (finalisé ou non).

Le consentement de l'utilisateur est représenté dans une structure de données distincte. L'objectif est d'avoir une structure de données qui enregistre les données relatives au consentement. Il est à noter que pour des fins de traçabilité, un consentement créé est ineffaçable même en cas de suppression du jeton lié. La structure de données d'un consentement comprend :

```
interface ERC721 {

//Events
event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId);

event Approval(address indexed _owner, address indexed _approved, uint256 indexed _tokenId);

event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);


//Functions
function balanceOf(address _owner) external view returns (uint256);

function ownerOf(uint256 _tokenId) external view returns (address);

function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external
     payable;

function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;

function transferFrom(address _from, address _to, uint256 _tokenId) external payable;

function approve(address _approved, uint256 _tokenId) external payable;

function setApprovalForAll(address _operator, bool _approved) external;

function getApproved(uint256 _tokenId) external view returns (address);

function isApprovedForAll(address _owner, address _operator) external view returns (bool);

}
```

Figure 3.1 API de ERC721

- le hachage du jeton,
- l'adresse du propriétaire du jeton,
- l'adresse de l'utilisateur auquel le consentement est donné (un processeur de données),
- le nombre d'accès autorisés.

De plus, un consentement contient une variable booléenne "active" qui indique si le consentement est actif ou expiré. L'expiration est causée par l'achèvement de nombre d'accès autorisés ou l'annulation du consentement par le propriétaire de données.

La figure 3.3 présente notre API de jeton conforme au RGPD qui contient les principales fonctionnalités de notre modèle :

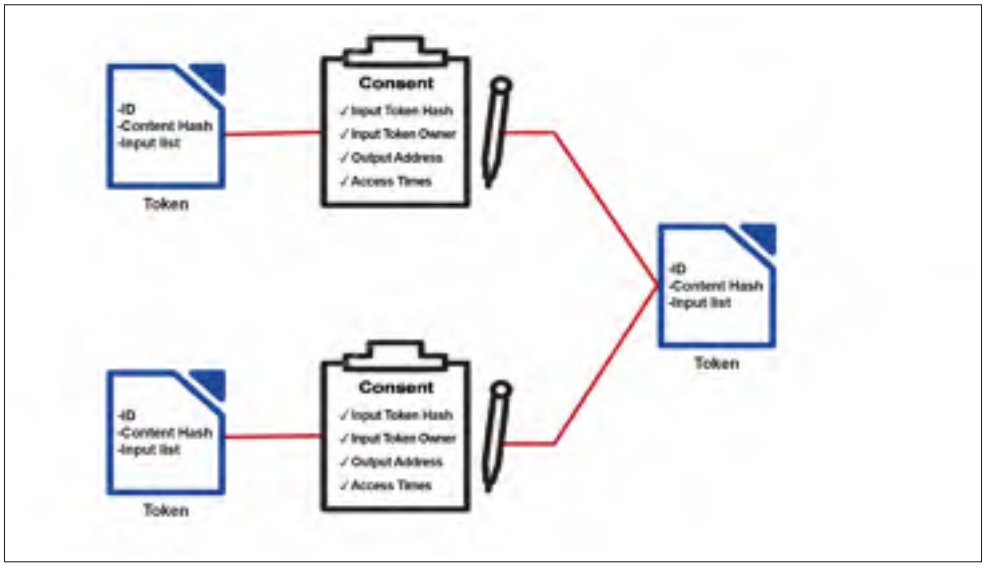

Figure 3.2 Modèle de tokenization de données

- **createToken** génère un nouveau jeton et l'ajoute à la liste des jetons. Après cela, il déclenche l'événement *TokenCreated* ;
- **viewToken** renvoie les attributs du jeton à son propriétaire ou à un utilisateur qui possède un consentement d'accès ;
- **viewTokensList** renvoie la liste des jetons appartenant à un certain utilisateur ;
- **addDataToToken** ajoute du contenu à un jeton existant par son propriétaire tant que le jeton n'est pas finalisé et déclenche l'événement *DataAdded*. Une fois finalisé, le jeton ne peut plus être mis à jour ;
- **addOutputToToken** ajoute un output au jeton identifié par *_tokenID*. Pour ce faire, la fonction finalise le jeton représenté par son *_tokenID* et génère un nouveau consentement contenant le *_tokenID*, l'adresse du propriétaire du jeton , *_outputAddress*, le hachage du jeton et la variable *_access* qui représente le nombre de fois que l'adresse output peut accéder au jeton. Ensuite, il génère l'événement *OutputAdded*. Une fois finalisé, un jeton ne peut plus être modifié ;
- **addInputToToken** ajoute un input identifié par *_inputTokenID* à un jeton identifié par *_tokenID*. Pour ce faire, la fonction vérifie si un consentement incluant *_inputTokenID* et

l'adresse du propriétaire de *_tokenID* existe et vérifie si $_access > 0$. Si ces conditions sont remplies, la fonction ajoute le jeton avec l'identifiant *_inputTokenID* en tant qu'entrée au jeton avec l'identifiant *_tokenID*. Il déclenche ensuite l'événement *InputAdded* ;

- **deleteToken** rend le jeton " indisponible " et déclenche l'événement *TokenDeleted*.

```
interface PrivacyCompliantToken{

//Events
event TokenCreated(uint256 ID);

event DataAdded(uint256 ID, string _data);

event OutputAdded(uint256 consentID, uint256 tokenID, address owner, address output);

event InputAdded(uint256 ID, address owner, int8 times);

event TokenDeleted(uint256 ID);


//Functions
function createToken(string calldata _data) external;

function viewToken(uint256 _ID, int256 _consentID) public view onlyAuthorized (int256(_ID),
    _consentID, 1) returns (uint256 ID, string memory _data, uint256[] memory _inputID, bool
     _available);

function viewTokensList() public view returns (uint256[] memory);

function addDataToToken(string calldata _data, uint256 _ID) external onlyOwner(_ID);

function addOutputToToken(uint256 _tokenID, address _outputAddress, int8 _access) external
    onlyOwner(_tokenID);

function addInputToToken(uint256 _tokenID, uint256 _inputTokenID) external onlyOwner(
    _tokenID);

function deleteToken(uint256 _ID) external onlyOwner(_ID);

}
```

Figure 3.3 API du jeton conforme à la confidentialité

3.3 Les interactions du systèmes

Le processus de collecte de données passe par plusieurs étapes qui demandent l'interaction avec notre contrat intelligent et l'IPFS. Nous montrons comment notre modèle fonctionne en utilisant un scénario simple : *Alice* ajoute son fichier de données au système et *Bob* traitera ces données pour générer des résultats. Pour ce faire, *Bob* doit également créer son propre jeton pour héberger

les résultats et le lier au jeton d'Alice pour représenter le flux de traitement des données. Cela n'est possible que si *Bob* obtient le consentement de *Alice* pour accéder à son jeton.

Comme étape primordiale, Bob doit générer un couple de clés privée/publique. Ensuite, il doit donner sa clé publique à Alice.

La figure 3.4 montre les différentes étapes suivies :

- *Création de jeton : Alice* télécharge ses données, chiffrées avec la clé publique de Bob, sur IPFS (1a), les épingle afin d'assurer leur persistance et obtient leur hachage en retour (1b). Ensuite, elle appelle *createToken* pour créer un nouveau jeton avec le hachage des données (1c). *Bob* suit l'étape (2) pour créer son jeton ;
- *Addition d'output : Alice* ajoute une output à son jeton en appelant *addOutputToToken* (3). Cette fonction finalise le jeton (c'est-à-dire que les attributs du jeton deviennent non modifiables) et crée un consentement qui autorise Bob à utiliser ce jeton d'Alice ;
- *Addition d'input : Bob* ajoute une input à son jeton en appelant *addInputToToken* (4) avec le jeton d'Alice en entrée. Cette fonction vérifie que le jeton spécifié contient un consentement actif à l'adresse de l'appelant. La réussite de cette étape permet à Bob de démontrer que les résultats contenus dans son jeton ont été générés à l'aide des données d'Alice et qu'il a obtenu son consentement pour utiliser les données de son jeton ;
- *Accès au jeton :* Pour afficher un contenu de jeton, *Bob* appelle la fonction *viewToken* (5a) pour accéder au hachage des données stockées sur IPFS. Le contrat intelligent trouve le consentement qui autorise Bob à accéder au jeton d'Alice et retourne son contenu (5b). Suite à cela, Bob récupère le hachage qui va lui permettre de demander les données auprès de IPFS (5c), et de les récupérer (5d) ;
- *Stockage du résultat :* Pour ajouter le résultat du traitement de données d'Alice, *Bob* doit suivre les étapes dans (6a), (6b), et (6c) pour ajouter le hachage du résultat à son jeton.
- *Suppression de jeton :* Pour supprimer son jeton, *Alice* appelle la fonction *tokenDelete* (6) qui rend le jeton indisponible pour lecture. Ensuite, elle détache les données de l'IPFS afin d'être supprimées dans la prochaine phase de ramassage des miettes.

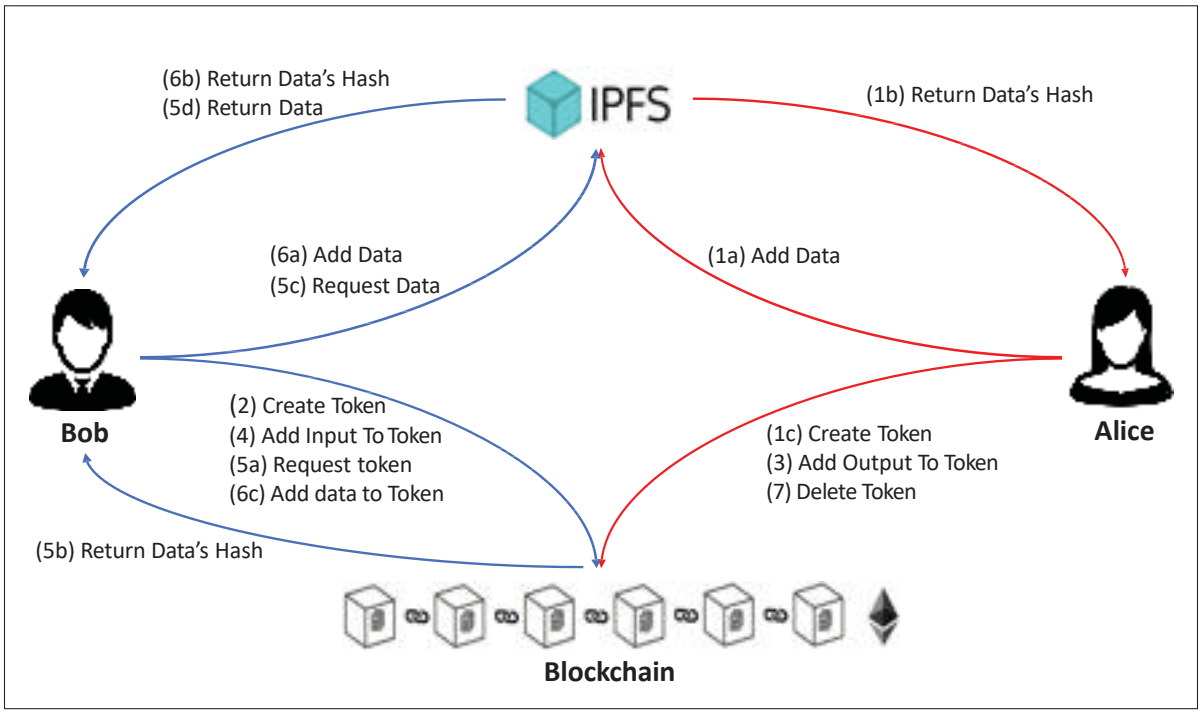

Figure 3.4 Interaction des acteurs du système

3.4 Les composants de l'architecture

La figure 3.5 montre notre architecture de système de référence. En plus de la chaîne de blocs Ethereum et de l'intégration avec IPFS, l'architecture contient plusieurs composants pour l'application client Web :

- **Web3.js :** Il s'agit d'un ensemble de bibliothèques qui fournit une API Ethereum JavaScript qui permet à l'utilisateur d'interagir avec le réseau Ethereum et de gérer les comptes Ethereum ;
- **MetaMask :** Il agit comme un pont entre le réseau Ethereum et l'application Web en injectant Web3 dans l'application et en hébergeant le portefeuille Ethereum dans le navigateur client.

Sur la base de cette configuration, lorsqu'un utilisateur souhaite exécuter une fonction de contrat intelligent, Web3 convertit cette demande en une transaction Ethereum qui sera signée et envoyée

par MetaMask. Ces transactions interagissent avec le contrat *PrivacyCompliantToken* qui est écrit en Solidity, compilé et déployé sur Ethereum.

De plus, l'application Web inclut **Ipfs-http-client.js** qui est une API Javascript permettant d'interagir avec un nœud IPFS qui permet à l'utilisateur d'exécuter toutes les fonctions IPFS, y compris l'ajout, la récupération et la suppression de fichiers.

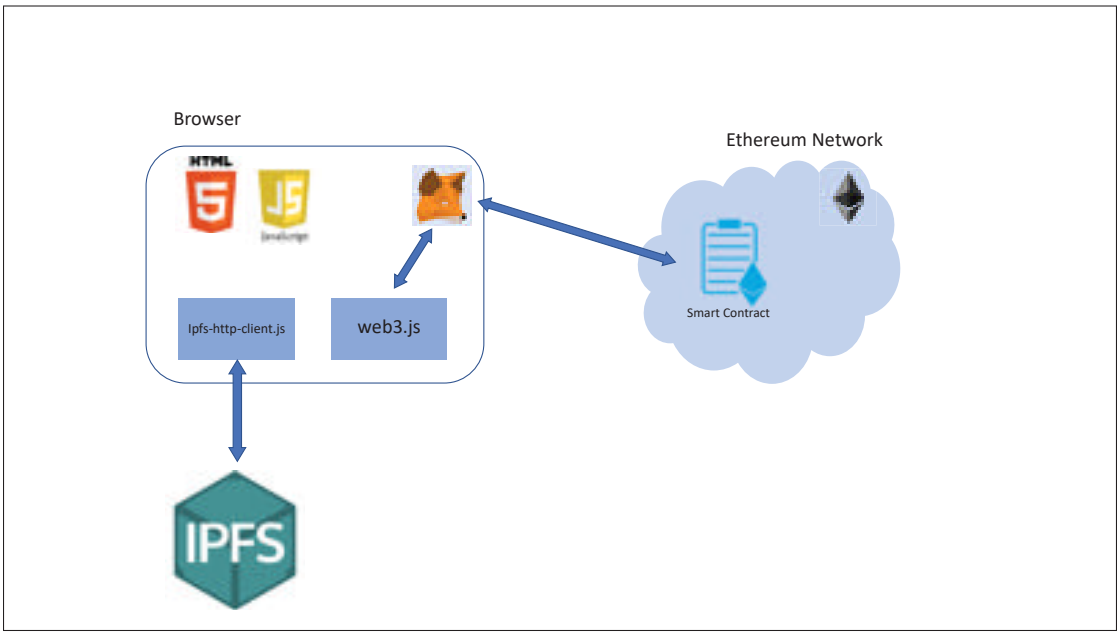

Figure 3.5 Architecture du système

3.5 Conclusion

Dans ce chapitre, nous avons analysé la norme ERC721 et nous avons démontré qu'elle n'est pas compatible avec les fonctionnalités souhaitées de notre système. Puis, nous avons présenté notre modèle de tokenisation de données et nous avons détaillé les fonctions principales de notre système de collecte de données. Ensuite, nous avons proposé un scénario de base pour expliquer les interactions dans notre système de collecte de données. Finalement, nous avons présenté les composants de l'architecture de notre solution.

CHAPITRE 4

ANALYSE DE LA CONFORMITÉ AU RGPD

Dans ce chapitre, nous expliquons comment notre modèle de tokenisation des données et notre architecture de référence proposés prennent en charge le RGPD. Nous démontrons en outre la conformité au RGPD à travers un exemple de scénario pour la formation en machine learning.

4.1 Prise en charge des droits RGPD

Le système de collecte de données que nous avons proposé est axé sur le respect du règlement européen RGPD d'une part et sur la traçabilité rigoureuse qui va permettre d'imposer le respect des règlements d'autre part. Notre travail est focalisé sur les droits des propriétaires de données qui sont interprétés dans la section 1.3.3. Notre solution proposée prend en charge les réglementations RGPD comme suit :

- **le droit de restreindre et de contrôler l'accès aux données :** Ce droit est accordé par la structure de données *Consent* qui contrôle qui a le droit d'accéder aux données personnelles et limite le nombre d'utilisations (Temps d'accès) ;
- **le droit de supprimer les données personnelles :** Toutes les données collectées doivent être téléchargées et épinglées à la couche de stockage hors chaîne *IPFS*. L'effacement est fourni par notre mécanisme de suppression qui consiste à désépingler le contenu d'IPFS et à rendre le jeton en chaîne indisponible. Une fois désépinglé, le contenu supprimé avec le ramasse miette et ne pourra pas être téléchargé. Le seul élément qui est gardé dans le jeton est le hachage du contenu, qui est nécessaire à des fins d'auditabilité. Nous soutenons que le maintien du hachage en chaîne ne viole pas notre interprétation du RGPD car le contenu original ne peut pas être reconstruit. En outre, le contenu non épinglé disparaît généralement d'IPFS dans les 24 heures (Steichen, Fiz, Norvill, Shbair & State, 2018), ce qui est inférieur au seuil d'un mois requis par le RGPD. Cependant, un acteur malveillant pourrait intentionnellement garder un certain fichier disponible même lorsque les récompenses d'épinglage ont disparu. Notre

interprétation fait donc valoir qu'une garantie probabiliste que les données seront effacées est suffisante pour satisfaire au RGPD en raison du manque d'incitations économiques à maintenir les données disponibles. Des mécanismes de suppression de contenu plus explicites pour le RGPD, tels que la liste noire [1], seront explorés dans les travaux futurs. Notez que nous interprétons également le droit d'effacement comme n'affectant que les données personnelles des utilisateurs, et ne pas exiger la suppression des résultats impliquant ces données. Ce dernier est également laissé comme travaux futurs. Également, la suppression de données ne concerne que les données qui existent sur le système. C'est à dire, notre mécanisme ne supprime pas les copies de données personnelles qui existent localement sur les machines des utilisateurs ;

- **le droit de transfert :** En utilisant notre système, un propriétaire de données peut demander ses données à tout moment. Pour ce faire, l'utilisateur peut consulter le contenu de ses propres jetons, ce qui est vérifié par le contrat intelligent. Ensuite, l'utilisateur récupère les fichiers de données d'IPFS en utilisant les hachages des jetons ;
- **Suivi de la piste d'audit de l'utilisation des données personnelles :** Nous considérons la chaîne de blocs comme l'enregistrement faisant autorité des activités de traitement des données par les processeurs de données. Tous les résultats générés par les processeurs de données doivent être capturés dans des jetons de données, les entrées appropriées faisant référence aux jetons contenant les données source utilisées pour générer ces résultats. Au cours d'un processus d'audit, une entreprise peut être arrêtée et condamnée à une amende pour avoir collecté des données qui n'ont pas été retracées sur la chaîne, produisant des résultats qui ne correspondent pas aux hachages des résultats enregistrés, répertoriant de manière incorrecte les données d'entrée utilisées pour produire un jeton de résultat, ou continuer à stocker des données pour les jetons supprimés. Notez qu'un processeur de données pourrait également fournir ses propres résultats à une troisième société, créant ainsi des chaînes plus longues de jetons de données liés, qui enregistrent la provenance de bout en bout.

[1] https://github.com/decentraland/ipfs-node/issues/14

4.2 Étude de cas d'utilisation : formation des modèles de machine learning

Afin de donner une utilisation concrète de notre système et de démontrer comment la conformité au RGPD est vérifiée, nous décrivons un cas d'utilisation de service mobile illustré dans la figure 4.1 : afin de **fournir une qualité de service adaptative** et d'**étudier les plans mobiles appropriés des clients**, un fournisseur de services mobiles utilise son réseau cellulaire pour collecter les données des clients (âge, sexe, position géographique ...) afin de former deux modèles d'apprentissage automatique qui suivent les modèles d'utilisation des services mobiles et fournissent des informations pour les objectifs susmentionnés.

Les clients téléchargent leurs fichiers d'informations personnelles sur IPFS, les épinglent et créent des jetons contenant le hachage des fichiers. Le fournisseur de services mobiles crée un jeton pour chaque modèle, puis ajoute les résultats de la formation des deux modèles.

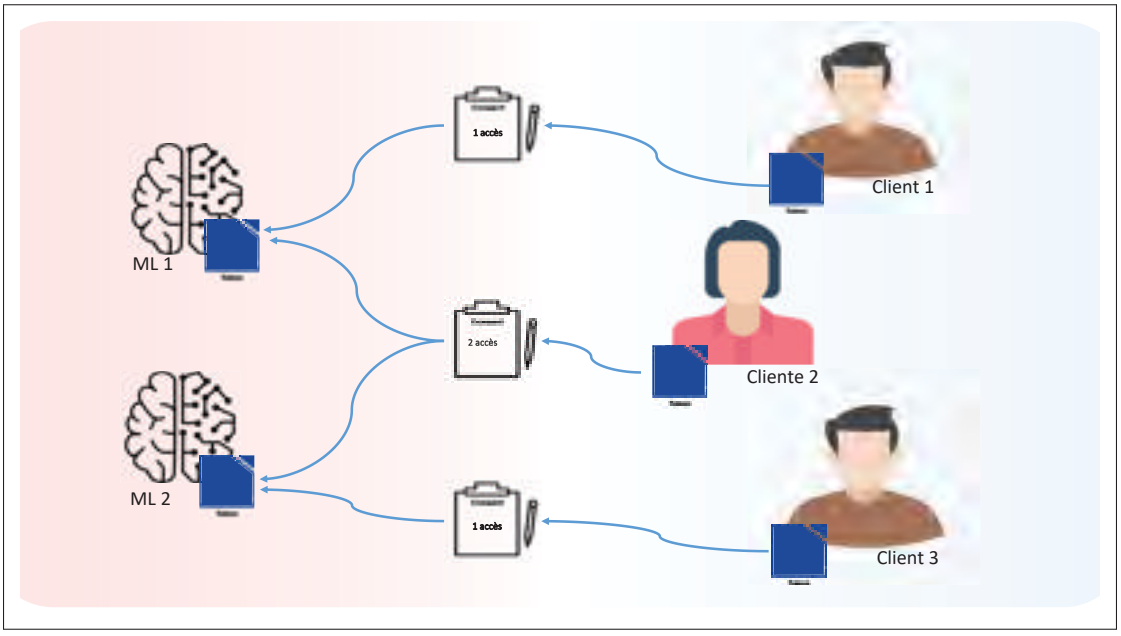

Figure 4.1 Collecte de données pour l'apprentissage automatique

Pour accorder l'accès à ses données, un client doit créer un consentement avec l'adresse du fournisseur de services mobiles en sortie. Le client a deux options :

- **option 1 :** Le client peut fournir un temps d'accès. Cela permettra au fournisseur de services mobiles d'utiliser les données du client pour l'un de ses modèles d'apprentissage automatique et l'empêchera de les utiliser pour les deux ;
- **option 2 :** Le client peut fournir deux temps d'accès. Cela permettra au fournisseur de services mobiles d'utiliser les données client pour les deux modèles.

La dernière étape de l'autorisation consiste à ajouter les jetons client en input aux jetons des modèles du fournisseur. De cette façon, le fournisseur de services mobiles sera autorisé à utiliser les données personnelles pour former les modèles conformément aux réglementations RGPD. Un client peut toujours arrêter l'accès aux fichiers de données personnelles en déclenchant le processus de suppression.

Étant donné que RGPD donne le droit à l'autorité de contrôle d'accéder aux données confidentielles à des fins de d'audit (article 58), l'autorité de contrôle peut suivre l'approche suivante afin de vérifier l'intégrité du modèle d'apprentissage automatique et de vérifier si les données d'entrée ont fait l'objet d'un consentement préalable ou non :

- rassembler les données d'entrée du jeton du modèle d'apprentissage automatique,
- former le modèle avec les données d'entrée collectées,
- comparer les résultats de la formation avec les résultats existants,
- Si nous obtenons des résultats égaux, nous pouvons conclure que le modèle a été formé avec les données exactes collectées avec consentement. Sinon, on conclut que le modèle a été formé avec des données obtenues illégalement.

Il est à noter que dans le cas où l'autorité de contrôle n'a pas le pouvoir d'accéder aux données personnelles, une couche cryptographique (Zero Knowledge Proof) pourrait être ajoutée au système pour prouver l'existence des données sans en révéler le contenu. Un tel mécanisme sera étudié dans le cadre de travaux futurs.

Pour valider la suppression d'un jeton, l'auditeur doit confirmer qu'il n'est pas possible d'accéder à son contenu depuis le système. Pour ce faire, l'auditeur demande le hachage IPFS auprès du jeton supprimé qui est conservé à cette fin, puis il essaie de demander les données à l'IPFS. Si les données sont effectivement supprimées, l'auditeur ne devrait pas recevoir de réponse du réseau.

Pour confirmer le respect du bon déroulement des opérations pour obtenir l'accès aux données personnelles des clients, l'auditeur peut suivre cette approche :

- sélectionner un jeton d'entrée dans le jeton du modèle d'apprentissage automatique,
- obtenir la hauteur des blocs qui contiennent les transactions qui créent la sortie pour le jeton de données client et les transactions qui créent l'entrée pour le jeton de modèle d'apprentissage automatique,
- Dans le cas normal, la hauteur du bloc de création d'output (création de consentement) doit être inférieure à la hauteur du bloc de création d'input,
- répéter les opérations précédentes avec les jetons d'input restants.

Enfin, pour valider le droit de transfert, l'auditeur peut demander le hachage de certaines données de son jeton et utiliser le hachage pour récupérer les données de l'IPFS. L'auditeur doit pouvoir obtenir une copie des données.

4.3 Conclusion

Dans ce chapitre nous avons étudié la conformité de notre modèle de tokenisation avec RGPD. En particulier, nous avons discuté comment le droit de restreindre d'accès, le droit de supprimer les données et le droit de transférer les données sont offerts par notre système de collecte de données. Par la suite, nous avons étudié un cas d'utilisation de notre système et nous avons démontré comment on peut vérifier le respect des droits des propriétaires de données.

CHAPITRE 5

ÉVALUATION DE LA SOLUTION PROPOSÉE

Dans ce chapitre, nous évaluons expérimentalement notre système de collecte de données. Pour ce faire, nous créons différents scénarios synthétiques. Ces tests visent à évaluer les performances du système en termes de consommation de gaz et de latence globale. Par la suite, nous investiguons l'apport de l'IPFS en terme de rentabilité par rapport au stockage en chaîne. Finalement, nous proposons un modèle d'utilisation le moins coûteux de notre système.

5.1 Configuration expérimentale

Nous utilisons la configuration d'évaluation suivante : les scénarios sont scriptés dans NodeJS et exécutés sur un ordinateur portable équipé d'un processeur Intel (R) Core (TM) i7 1,90 GHz fonctionnant sous Windows 10. Le contrat intelligent est déployé sur un réseau privé Ethereum Ganache. Nous décidons d'utiliser 16 gwei comme prix du gaz, qui est la moyenne au moment de l'expérience selon *Etherscan*, avec un temps de confirmation moyen de 37 secondes. Dans chaque scénario, 100 opérations sont exécutées par 3 utilisateurs.

5.2 Effet du taux de création de jetons

Dans cette expérience, nous évaluons l'effet du taux de création de jetons sur la consommation totale de gaz. Nous exécutons le système huit fois avec un nombre fixe d'opérations (100 opérations), y compris des opérations de création de jetons aux côtés d'autres opérations. À chaque exécution, nous augmentons le taux de création de jetons par rapport à d'autres opérations. Comme le montre la figure 5.2, l'augmentation des créations de jetons augmente linéairement avec la consommation globale de gaz. Nous mesurons la consommation de gaz de chaque opération et constatons que *createToken* coûte 276476 unités de gaz (figure 5.1), ce qui est supérieur à la consommation moyenne de gaz des autres opérations de 102355 unités de gaz. Cela est dû au coût de stockage d'une nouvelle structure de données en chaîne ainsi qu'à la complexité de la fonction *createToken* en termes de nombre d'opérations.

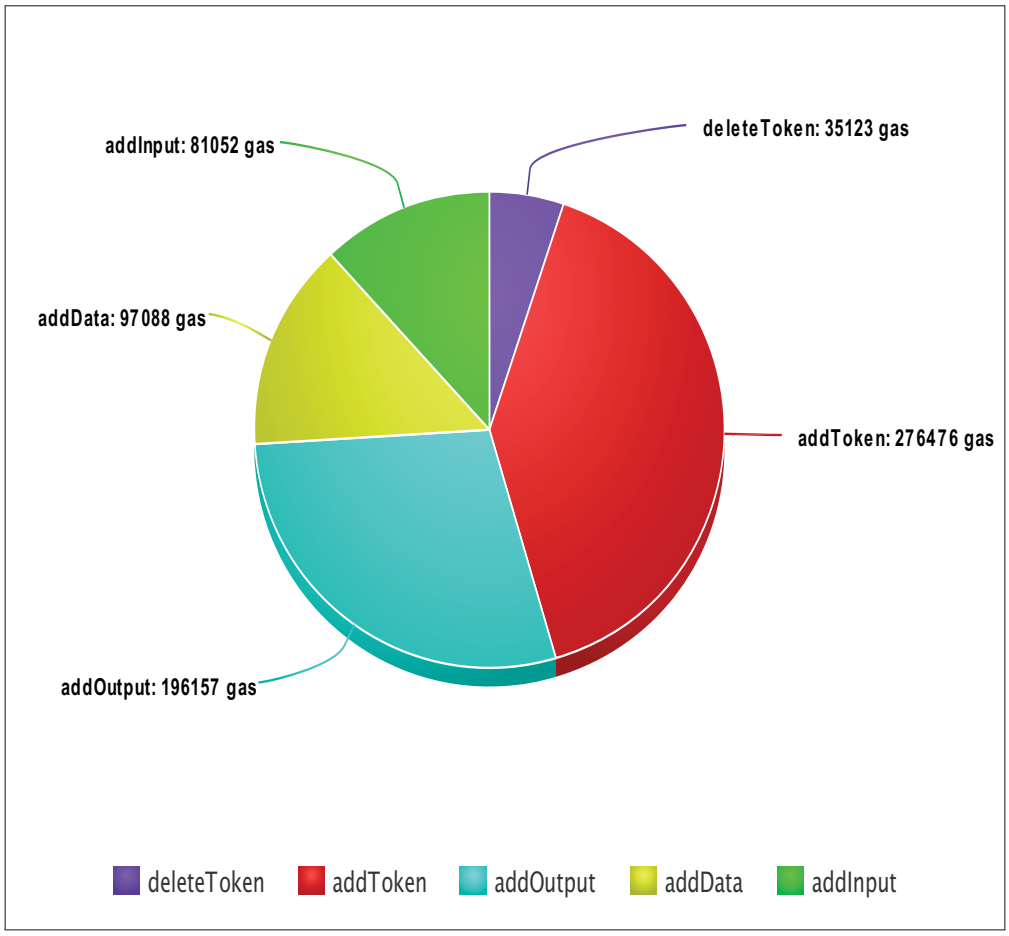

Figure 5.1 Répartition de la consommation de gaz

5.3 Effet du taux de output par jeton

Dans cette expérience, nous évaluons l'effet de l'augmentation du nombre d'output par jeton sur la consommation globale de gaz. Nous exécutons le système 4 fois avec 100 opérations, dont 20 *createToken*. Pour chaque itération, nous augmentons le nombre d'output par jeton et mesurons la consommation globale de gaz de chaque exécution. Comme le montre la figure 5.3, l'augmentation du nombre d'output par jeton est suivie d'une augmentation significative de la consommation de gaz. Cela est dû au coût de *addOutput* qui s'élève à 196157 unités de gaz (Figure 5.1). La raison de cette consommation de gaz est la création d'une nouvelle structure de données de consentement au stockage de contrat intelligent.

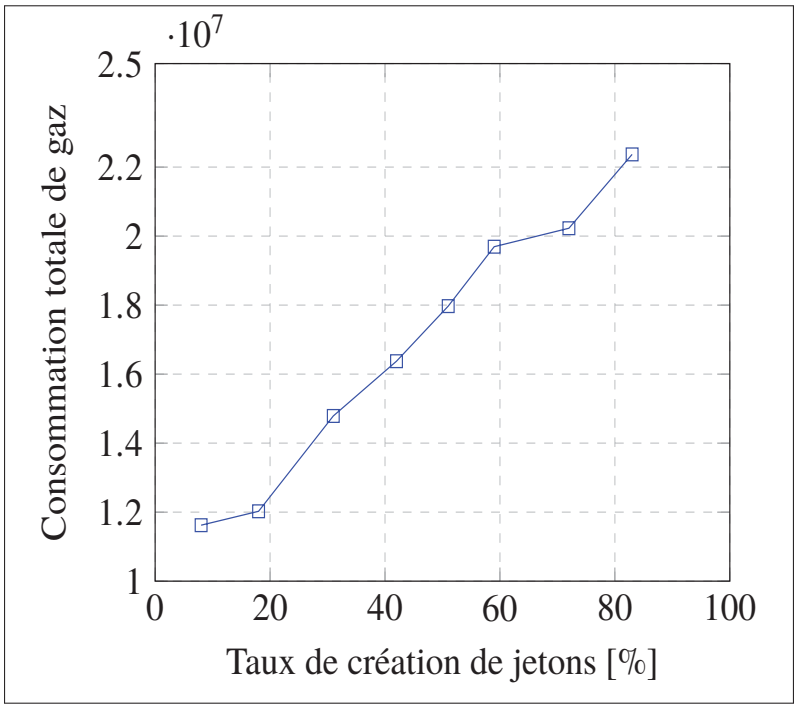

Figure 5.2 Consommation totale de gaz par taux de création de jetons

5.4 Effet du taux de input par jeton

Dans cette expérience, nous évaluons l'impact de l'augmentation de l'ajout d'inputs par jeton sur la consommation globale de gaz. Nous lançons le test 4 fois avec 100 opérations dont 20 *createToken*. Pour chaque itération, nous augmentons le nombre d'input par jeton et mesurons la consommation totale de gaz. La figure 5.4 montre une diminution de la consommation de gaz qui est due au coût de la fonction *addInput* qui est égal à 81052 unités de gaz. Cela est dû à la simplicité des instructions de *addInput*.

5.5 Effet du volume de données

Le but de cette expérience est de déterminer l'impact de la taille des données sur la latence du système. Nous générons différents fichiers factices de tailles différentes et pour chaque taille, nous exécutons l'expérience en utilisant la même configuration : nous générons 100 opérations,

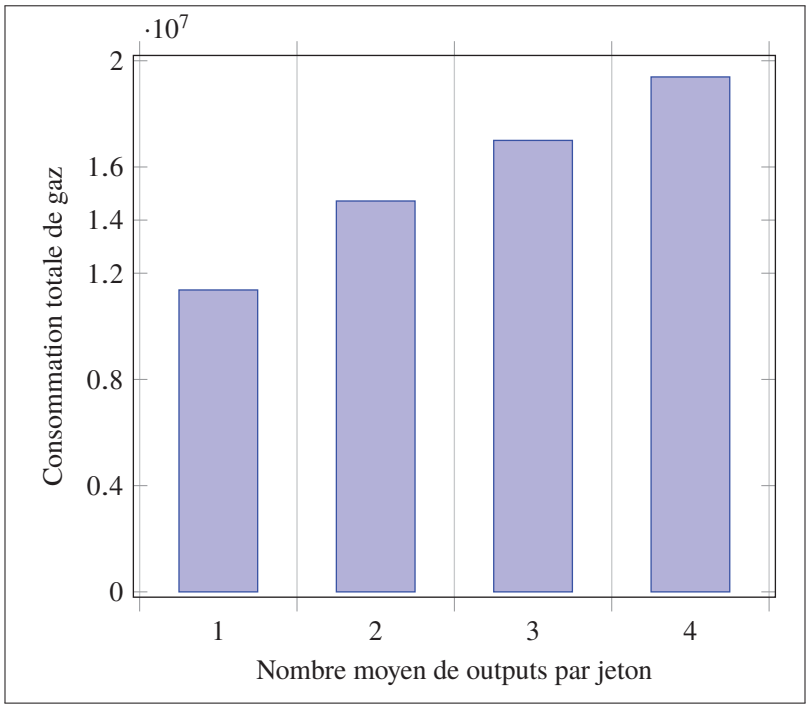

Figure 5.3 Consommation totale de gaz par output moyen par jeton

dont 18 *createToken* et 18 *viewToken*. Étant donné que Ganache ne prend pas en compte le délai de confirmation de transaction, nous définissons un délai fixe qui est égal à 37 secondes. Cette valeur correspond au délai de confirmation moyen pour un prix du gaz égal à 16 gwei au moment de l'expérience selon *Etherscan Gas Tracker*. Comme le montre la figure 5.5, la latence globale du système est presque constante ($\sim$62.47min). Cela prouve que la taille des données n'affecte pas la latence du système.

5.6 Rentabilité de IPFS

Pour démontrer la rentabilité de l'IPFS par rapport au stockage en chaîne, nous étudions le coût de stockage des données sur un stockage de contrat intelligent qui peut être considéré comme un large éventail de " mots " de 32 octets. Selon (Wood et al., 2014), **SSTORE** est l'opcode chargé de stocker un mot sur le stockage et coûte 20000 gaz. Avec le coût du gaz considéré de 16 gwei et le taux de conversion de 165,52 USD pour 1 Eth, nous calculons le coût de stockage de

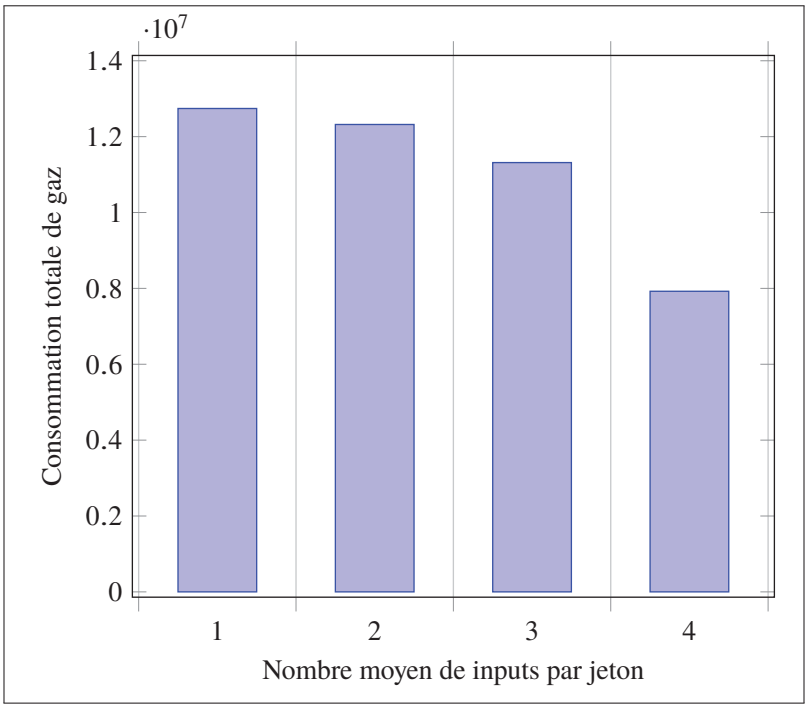

Figure 5.4 Consommation totale de gaz par input moyen par jeton

différentes quantités de données en chaîne. Les résultats collectés présentés dans le tableau 5.1 montrent que le coût de stockage d'une petite quantité de données sur le stockage de contrat intelligent est très élevé (par exemple, le stockage de 100 kilo-octets coûte 165,52 USD). En revanche, l'utilisation d'un service d'épinglage IPFS est beaucoup moins chère. Par exemple, "Pinata" offre un service gratuit lorsque la quantité de données est inférieure à 1 Go et des frais de 0,15 USD/mois pour les données supérieures à 1 Go. Bien que le stockage en ligne nécessite un paiement unique, il est toujours moins cher et plus pratique de stocker des données sur IPFS.

Tableau 5.1 Coût de stockage sur-chaîne

Taille des données(Koctets)	Coût de stockage(gaz)	Coût de stockage(ETH)	Coût de stockage(USD)
1	640000	0.01	1.69
10	6260000	0.1	16.57
100	62500000	1	165.52
1000	625000000	10	1655.20

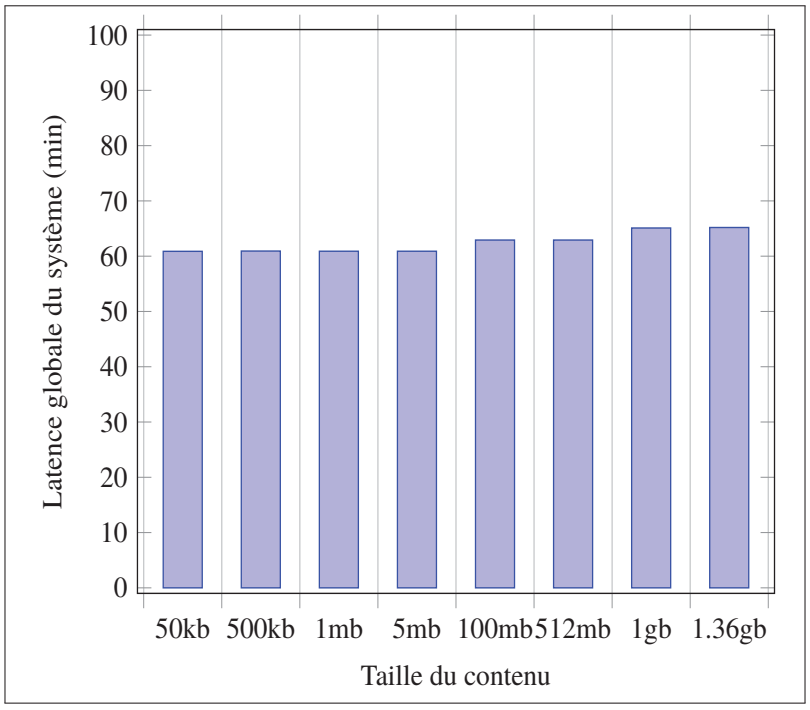

Figure 5.5 Impact de la taille du contenu sur la latence du système

5.7 Discussion

Nous observons que les fonctions *createToken* et *addOutput* consomment une quantité importante de gaz. Avec le même coût de gaz et le même taux de conversion que ceux utilisés pour les expériences, les frais de transaction en gwei pour les fonctions de création et d'ajout de sortie sont respectivement de 4423616 (0,73 USD) et 3138512 (0,52 USD). Selon Etherscan, les frais de transaction moyens sont de ~0.31 USD. Notez que les frais peuvent être réduits en réduisant le prix du gaz en échange d'une augmentation du temps de confirmation. Par exemple, si nous définissons un 8 gwei / gaz, les frais seront de 0,37 USD et le temps de confirmation pourrait aller jusqu'à 3 minutes.

Selon les résultats, nous soutenons qu'un modèle d'utilisation optimisé de notre système utilise une quantité limitée de jetons, une faible quantité de création d'output, mais un nombre plus élevé d'input par jeton. Notre système évolue bien avec des fichiers de grande taille. L'exemple décrit

dans la section 4.2 est donc un cas d'utilisation approprié du point de vue des performances. Pour les clients n, 1 jeton et 2 sorties seront créés par chaque client, et 2 jetons et $2 \times n$ entrées seront créés par le fournisseur de services mobiles.

CHAPITRE 6

EXPÉRIENCE PERSONNELLE

Le projet *Un modèle de tokenisation basé sur la chaîne de blocs pour la traçabilité et la conformité de la collecte de données* fait partie d'un programme de maîtrise avec mémoire d'une durée de deux ans, effectuée à l'École de technologie supérieure. Pour effectuer nos objectifs, nous avons dû passer par plusieurs étapes et surmonter plusieurs défis.

Au cours de cette période, nous avons commencé par définir la proposition de recherche, fixer les objectifs et les lier à une chronologie. Ensuite, nous avons passé en revue les documents qui expliquent les notions de base relatives à notre projet et les articles qui décrivent les œuvres connexes. Par la suite, nous avons commencé à nous familiariser avec les outils de travail tels que le langage de programmation des contrats intelligents "Solidity", l'environnement de développement "Truffle" et l'interface de programmation "web3.js". La conception de la collecte de données fut la prochaine étape de notre projet. Nous avons défini le modèle de tokenisation de données et le système de fichier pour stocker les données personnelles. Nous avons passé par la suite à l'implémentation du projet. Nous avons développé le contrat intelligent, et les clients web. Ensuite, nous avons optimisé l'implémentation pour réduire le coût. Nous avons passé à l'évaluation de notre travail en implémentant des scénarios de test synthétiques. Parallèlement, nous avons commencé à rédiger un article scientifique pour publier notre travail dans une des conférences reconnues. Finalement, nous avons rédigé notre mémoire de maîtrise qui s'est basée sur notre article.

Pour réaliser cet œuvre, nous avons surmonté plusieurs défis. Premièrement, les normes de tokenisations existantes telles que ERC20 et ERC721 n'étaient pas adaptées à la collecte de données. C'est pourquoi nous avons décidé de mettre en place notre propre modèle de tokenisation. Deuxièmement, dans la phase de conception on a constaté que la chaîne de blocs n'est pas conçue pour supporter le stockage de données personnelles pour deux raisons. La première raison est lié au fait que la taille moyenne d'un bloc Ethereum est de 20 à 30kb [1]. La

[1] https://ethgasstation.info/blog/ethereum-block-size/

deuxième raison est lié à l'immuabilité de la chaine de blocs. Cela rend le droit de supprimer les données personnelles irréalisable. Pour remédier à cette limitation, nous avons pris la décision d'utiliser un système de fichier hors chaîne et garder un lien sur la chaîne pour accéder aux données.

À l'échelle personnelle, ce parcours fut une expérience très valorisante sur le niveau technique. Tout au long de notre projet, nous avons étudié plusieurs notions des systèmes distribués. En particulier, nous avons maîtrisé les systèmes à base de chaînes de blocs, les cryptomonnaies, et le développement des contrats intelligents. Au delà de l'aspect technique, ce projet de maîtrise m'a offert l'opportunité d'entrer dans le monde de la recherche, d'acquérir de nouvelles compétences telles que la gestion de temps, la pédagogie, le partitionnement des tâches et le travail en équipe.

CONCLUSION ET RECOMMANDATIONS

Avec l'application du nouveau RGPD, les droits des propriétaires de données ont été révisés, la façon dont les données confidentielles sont traitées a été mise à jour et de lourdes amendes sont appliquées aux entités qui ne respectent pas les règlementations. Ainsi, les organisations doivent utiliser de nouveaux mécanismes pour se conformer à ses règles de confidentialité. Dans ce mémoire, nous proposons une combinaison innovante d'un système de stockage de fichiers décentralisé comme IPFS et d'une chaîne de blocs DLT pour construire un système de collecte de données conforme à la confidentialité. Plus précisément, nous introduisons un nouveau modèle de tokenisation des données qui permet aux propriétaires de données de définir des politiques de consentement et de contrôler l'accès. Nous utilisons Ethereum comme un registre immuable qui enregistre les pistes d'audit des activités de traitement des données et comme une plateforme de contrat intelligent pour valider les opérations de données avec la conformité au RGPD. IPFS offre une disponibilité réglable, qui peut être exploitée pour prendre en charge le droit d'effacement.

Nous avons implémenté notre système à l'aide d'un contrat intelligent écrit en Solidity. Nous avons aussi évalué les performances de notre système en termes de consommation de gaz et de latences. Les résultats montrent que les fonctions de création de jeton et de création de consentement ont une consommation de gaz significativement plus élevée que les autres fonctions. De plus, la taille des données n'a aucun effet sur la latence du système. Nous soutenons qu'un ajustement du prix du gaz est une solution possible au coût du gaz. De plus, nous observons que le cas d'utilisation étudié de la formation en apprentissage automatique correspond bien au modèle d'utilisation optimal pour la performance.

Dans notre travail de recherche, nous avons démontré que la technologie des registres distribués est une bonne solution pour la collecte de données dû à son immuabilité et sa transparence.

Notre modèle de collecte de données basé sur la tokenisation garantit les droits des propriétaires de données conformément au RGPD tout en gardant une piste d'audit.

Il convient de noter que certaines directions de travaux futurs peuvent être explorées. Premièrement, notre système se concentre sur le RGPD. Cependant, avec l'apparition des nouvelles lois comme la loi sur la protection des renseignements personnels des consommateurs de Californie (CCPA) ou la loi générale brésilienne sur la protection des données (LGPD), une extension du spectre d'applicabilité pour adresser ces nouvelles réglementations est envisageable. Deuxièmement, dans notre implémentation nous avons utilisé un système de fichier décentralisé proposant une suppression de données qui n'est pas radicale. C'est pourquoi une amélioration au niveau du stockage des données confidentielles constitue un des travaux futurs. Finalement, nous comptons optimiser notre modèle de tokenisation de donnée pour minimiser le coût de certaines fonctions telles que l'addition d'un nouveau jeton de données, et l'addition d'un output à un jeton existant.

ANNEXE I

ARTICLE SOUMIS

Cet article, intitulé "A Blockchain-Based Tokenization Model for Data Collection Traceability and Compliance", est soumis à Crypto Valley 2020, 3RD IEEE CRYPTO VALLEY CONFERENCE ON BLOCKCHAIN TECHNOLOGY (June 2020, Zug-Rotkreuz).

A Data Tokenization Model for GDPR Privacy Compliance

Alaeddine Chouchane, Kaiwen Zhang, and Chamseddine Talhi
Department of Software & IT engineering
École de Technologie Supérieur (ÉTS)
Montreal(QC), Canada

***Abstract*—As a new privacy regulation, the European General Data Privacy Regulation (GDPR) will disrupt the data collection industry which must comply with its clauses regarding transparency, traceability and data governance. In particular, the GDPR confers three important rights to individuals: the right to erasure, the right to data portability and the right to restrict processing. In this paper, we leverage the immutability offered by distributed ledger technologies (DLTs), combined with transparent and reliable validation logic execution via smart contracts, to support GDPR-compliant data collection. We propose a novel data tokenization format, which records consent and provide an audit trail capturing the data processing flows. We show how our data tokenization approach can be combined with a decentralized file storage system (such as IPFS) in order to efficiently store large volumes of data. We also demonstrate how data deletion can be supported by manipulating the data availability mechanism provided by IPFS. We describe a case study for machine learning training, showcasing the ability of our model to comply with GDPR. We implement our model using Solidity, provide an evaluation of our system performance using Ethereum and IPFS, and conduct a sensitivity analysis of the main functions provided by our decentralized application.**

***Index Terms*—DLT, Privacy, GDPR, Smart Contract**

I. Introduction

Privacy is one of the most challenging issues surrounding data collection as it is notoriously difficult to detect and prevent data leaks [1]. Statistics published by the Breach Level Index shows that more than 3 billion data records have been compromised in the first half of 2018[1]. As a solution to this problem, GDPR, a new data privacy regulation, came into effect in May 2018. The new European law is disrupting industries dealing with personal data who must comply to the regulation or face excessive fines which can reach 4% of the global company's revenues [2].

Nowadays, big companies are increasingly becoming data-driven. The data collection process involves obtaining streams of data from individual customers (called data owners), consuming the data to derive analytics and insights, and storing relevant information in a data warehouse. The data privacy compliance issues makes it necessary to provide data collection systems that allows verification for privacy breach in accordance to the consent policies given by data owners. Companies (data processors) must keep an audit trail of the data handling process. Data processors must also satisfy the data providers request to obtain a copy of their data, and to erase (forget) their information.

In light of these goals, we consider the use of distributed ledger technology (DLT) as a good candidate to assist in privacy compliance [3]. Blockchain DLTs are well suited for data log auditing due to its immutability which is provided by cryptographic mechanisms. Privacy regulation compliance can be supported by smart contracts, which are self-executing programs triggered at various steps of the data collection process. Finally, the transparency provided by blockchains and smart contracts allow all data owners and processors to understand and approve the data collection logic.

In this paper, we propose a novel data tokenization model to capture an audit trail of the data collection process and comply with GDPR regulations. We motivate the creation of our new model as established tokenization standards, such as ERC20 [4] and ERC721 [5], are not suitable for data collection purposes. Our data token model captures user consent, the relationship between source data and the resulting output, as well as the integrity of the data.

However, the blockchain itself is an incomplete solution for two reasons. First, immutability makes it unfeasible for data owners to delete their private data as required by GDPR regulations. Second, blockchains are not designed to store large volumes of data, as it would be case for Big Data applications. Therefore, we propose the use of a secondary off-chain storage layer to provide efficient and erasable data warehousing. In particular, we employ the InterPlanetary File System (IPFS) as a storage layer for our platform, which is a peer-to-peer file storing network featuring adjustable availability through replication, integrity through hash addressing, and cheaper hosting costs than on-chain solutions. We show how IPFS can be integrated to our blockchain solution to support the right of erasure required by the GDPR.

In this paper, we propose a decentralized data collection platform which complies with GDPR regulations regarding consent, data handling traceability, and erasure rights. Our contributions are as follows:

- We propose a novel model of data tokenization to represent the data collection and processing while ensuring traceability and consent (Section IV),
- We leverage IPFS as a secondary off-chain storage layer to support efficient data storage and the GDPR right of erasure (Section IV-B),

[1]https://www.thalesesecurity.com/2019/data-threat-report

- We show how GDPR compliance is achieved with our solution and describe case study for machine learning training (Section V), and
- We implement our solution as a smart contract using Solidity, and evaluate its performance using Ethereum (Section VI).

This paper now continues with the related works (Section II) and an overview on the technologies used in our solution (Section III).

II. Related Works

In this section, we review the two main categories of works related to our paper. First, we survey papers which leverages smart contract to preserve privacy and access control. Second, we look at papers focusing on data collection.

A. Privacy and Smart Contracts

Blockchain technology is employed in numerous research articles on data privacy and access control. FairAccess [6] is an access control framework that defines an Authorization Token as a currency that represents the access right given by the transaction's creator to its receiver. CreditCoin [7] is a blockchain-based incentive announcement protocol used for VANETs (Vehicular ad-hoc networks). It relies on cryptocurrency to encourage users to make traffic announcements. The privacy of the announcers is protected by a threshold ring signature mechanism. ShadowEth [8] and Hawk [9] present a privacy-preserving smart contract framework that relies on the TEE (Trusted Execution Environment) to handle private data. [9] splits the Hawk program into a private portion and a public portion while [8] deploys the contract in the trusted environment and keeps the contract identification in the public blockchain. [10] discusses how blockchain can be an alternative solution for privacy in the healthcare domain. The authors argue that blockchain can be used for access control for health records. Unlike these previous works, our research concentrates on privacy compliance from the GDPR perspective, and focuses on the rights of consent, transfer, and erasure.

In [11], the authors build a blockchain-based public ledger for policy compliance. Verification of compliance to the privacy policy is performed on an off-chain channel called VOCMC (Verifiable Off-Chain Message Channel) that involves two or more parties that need to agree over the policy compliance or violation based on their own version of the policy compliance evidence. The result of the agreement negotiation is then signed and stored on the blockchain. This process builds a public record of privacy compliance that enables organizations to build a trusted reputation when they stick to their privacy policies. The authors propose a scheme to verify privacy policy compliance using an abstract model of the GDPR regulations. In our work, we focus on the details of the GDPR regulations regarding the rights of data owners and we build a data collection system compliant with our interpretation of those rights.

B. Data Collection and Sharing

Many researches have addressed the data collection and sharing issue, especially in the cloud. Chu et. al. proposed in [12] an efficient data sharing in the cloud storage by creating key-aggregate cryptosystem (KAC) that enables sharing encrypted files which are stored in the cloud with access control by providing a single decryption aggregate key. The size of this key is constant and is independent from the number of files to share.

In [13], the authors addressed the lack of data owner control in cloud services by building an information accountability framework to keep track of the actual usage of user data. The framework enforces data access authorization based on user defined policies and two modes of logging for auditing and tracking usage history. Mona [14] is a scheme that provides multi-owner data sharing among dynamic groups in the cloud. This scheme enables user addition and revoking, data creation, data deletion and data access in a sharing group with consideration of a revoking list. In our work, we provide an alternative approach to traceability and access control using blockchains which is not necessarily specific to cloud usage.

Moreover, many articles exist on blockchain-assisted data collection and sharing. ProvChain [15] is an architecture that enables the collection and the validation of data provenance (i.e., the history of creation and operation executed on a cloud data product). Data provenance records are collected and stored in the blockchain to guarantee its integrity. In [16], authors created a user-centric data sharing scheme for mobile healthcare applications. Healthcare providers and insurance company request access from the owner to read health records. Health and medical data are hashed and stored in the blockchain to provide integrity. Access requests and activities are also recorded in the blockchain for auditing purpose.

Unlike these previous works, our work leverages smart contracts to define the data collection model and integrates blockchain and IPFS together to support the principal rights of the GDPR.

III. Background

In this section, we first review the major technologies necessary for our work, which are blockchains, smart contracts, Ethereum, tokenization standards, and IPFS. Then, we provide a summary of the GDPR and describe our interpretation of the regulations, which serve as the basis for the requirements supported by our solutions. Note that the GDPR and the interpretation of its rules may change over time; we leave it as future work to adapt our current solution to an evolving model of the GDPR.

A. Blockchain and Distributed Ledger Technologies

Serving as the backbone of Bitcoin, Ethereum, and other cryptocurrencies, a blockchain is a distributed ledger of records that records transactions replicated in a decentralized peer-to-peer network. Once in the ledger, transactions are permanent, verifiable, and tamper-proof. Transactions are collected together in blocks which are linked together

cryptographically to form the blockchain. In our work, we leverage DLT for the general purpose of data exchange beyond cryptocurrencies, thus enabling its usage in other domains such as health care, e-Government, and Internet of Things.

B. Ethereum and Smart Contracts

Ethereum provides a decentralized application layer on top of the blockchain to support operations beyond cryptocurrency [17]. This is enabled through smart contracts, which are autonomous programs executed by transactions [18]. Since the smart contract executable bytecode is itself stored as data on-chain, the immutability property of blockchains guarantee reliable execution of the smart contract program. Furthermore, the transparency of the executable bytecode allows for its integrity to be verified when compared against the compiled source code.

There are two types of accounts in Ethereum and each account has a 20 bytes address:

- **Externally owned account:** is controlled by an Ethereum user and has a public/private key pair.
- **Contract account:** is owned by a smart contract and is controlled by its code.

A contract account contains a combination of smart contract bytecode and contract-specific data and is deployed by a externally owned account to the blockchain. Smart contracts are written in Ethereum Virtual Machine code (EVM code) and provides an Application Binary Interface (ABI) that allows users to interact with it. A contract function call included in a transaction is executed by the miner who included the transaction inside a block and is verified by every node who gets a copy of the block.

In order to send coins to another account, to deploy a smart contract, or to execute the contract, an externally owned account creates a transaction which contains the following fields:

- **Nonce:** The number of transactions sent by the account. It increments every time this account sends a transaction.
- **Gas Price:** The price of a gas unit the user is intending to pay.
- **Gas Limit:** The maximum number of gas units the user allows to be used in this transaction.
- **Destination Address:** The address to which the transaction is sent.
- **Value:** The amount of Ether the user wants to send.
- **Signature:** The cryptographic signature of the transaction sender.

Gas is the unit to calculate the cost of every operation in Ethereum. This includes the execution of the operation codes (Opcodes) in a smart contract function and storing data in the account state. The gas price is calculated in ether and is freely defined by the transaction sender; however, users should define a high gas price in order to have to encourage miners to add their transactions rapidly to a block.

In our work, we employ the smart contracts to encode our data collection model and we use gas cost as a metric to evaluate the system performance and provide a sensitivity analysis of various parameters in order to understand the optimal usage pattern of our solution. While our work is currently implemented using Solidity and deployed on Ethereum, the proposed model is general and can be implemented in other languages. We chose Solidity as it is currently the most mature smart contract programming language and benefits from a strong development community and tool kits. Therefore, Ethereum is a natural choice as well as it supports Solidity natively. However, depending on the specific data collection application, a different blockchain system may be desirable, such as a permissioned one [3]. In this case, we can re-implement or adapt our Solidity code to work on a different platform. For example, Quorum supports Solidity code in a permissioned setting [19].

C. InterPlanetary File System

IPFS (InterPlanetary File System) was initially released in 2015 with the intention to build a decentralized file system infrastructure for the future web. IPFS combines elements of previously successful distributed systems, namely DHTs, BitTorrent and Git [20].

Since IPFS is based on a public and unreliable peer-to-peer network, the traditional location-based data addressing may lead to integrity issue since the validity of the content of a file is not checked. To overcome this problem, IPFS uses content-based addressing, which consists on calculating a hash of the file which will be used for lookup. This way, if we want to verify the integrity of a file we only have to recalculate its hash and compare it with the hash used as the address.

When a file is added to IPFS, it is stored in blocks on the IPFS repository of a node and these blocks are linked together in a Directed Acyclic Graph (DAG). To locate a content-addressed file, IPFS uses the *Kademlia* Distributed Hash Table (DHT) [21]. This mechanism can find the peerID that provides a certain block or a file. To optimize storage space, every IPFS node has a garbage collector that deletes periodically all *unpinned* blocks. Thus, if we want to guarantee the persistence of a file, we need to explicitly *pin* it. A file can later be *unpinned* in order to be deleted or garbage collected. Pinning services exist to guarantee availability of a certain file in exchange for a periodic cost.

IPFS is used in our system to store data because of the integrity properties provided by its content hashing scheme. Besides, it is used as a cheaper file storing alternative due to the high cost of persistent storage on the Ethereum blockchain. Finally, we can leverage the *pinning* mechanism to control availability of the data for GDPR compliance.

D. General Data Protection Regulation

GDPR (General Data Protection Regulation) is a new set of regulations for data privacy protection which took effect on 25 May 2018. GDPR is introduced in reaction to companies collecting a massive amount of sensitive information from European citizens without a clear explanation on how this data is processed.

With the new regulations, companies must raise the level of transparency of their data collection process. Data processors will be regularly audited for GDPR compliance and may receive sanctions that can reach up to 4% of annual global turnover. One important consequence of the GDPR is its extraterritorial applicability [2], which means that GDPR rules apply to organizations that process European data regardless of the geographical location of the data processors.

According to the Official Journal of the European Union [22], individuals (data owners) benefit from the following rights from the companies (data processors):

- The right to restrict and control data access: Data owners have the right to control data access by providing a clear and straightforward consent to the data processor. Furthermore, data owners have the right to limit and restrict the access given to a data processor.
- The right to erasure: Data owners have the right to delete their personal data so that they will be no longer accessible by the data processor. The request should be processed within one month of its receipt and without unjustified delays.
- The right to transfer: Data owners have the right to have their data transferred to them in a machine readable format and have the right to send it to another data processor.

E. Tokenization Standards

There exist already many standards that allow smart contract developers to create tokens. These standards provide mandatory functions to be implemented in order to create interoperable tokens. Among these standards, we find ERC20 which is a guideline for creating *fungible* tokens [4]. These token can take the form of cryptocurrencies, shares, or any other countable resource that is interchangeable and divisible. This model is not suitable for data collection due to its non-uniqueness and divisibility properties which make two tokens with the same value worth the same (a bill of $10 worth the same as another bill of $10). In our context, we wish to capture data from individual users inside of tokens which are uniquely identifiable, and therefore non-fungible.

ERC721 is another standard that supports non-fungible tokens [5]. These tokens represent assets which are unique, non-interchangeable and non-divisible like collectible items, diamond certificates, or identity documents. Since its release in September 2017, many blockchain-based game applications had been developed like *CryptoKitties* and *EtherTulips*. While ERC721 does satisfy the non-fungible requirement of data collection, it has other features which are not compatible, which we will investigate in the next section.

IV. Proposed Solution and Architecture

In this section, we first analyse ERC721 in the context of data collection. We then present our novel data tokenization model and a reference architecture deployed over Ethereum.

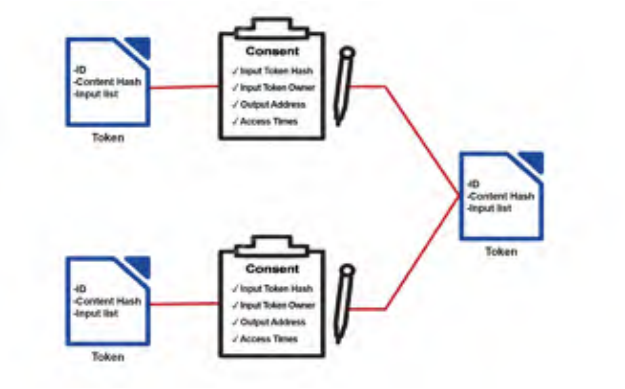

Fig. 1. Data Tokenization Model

A. Proposed Novel Data Tokenization Model

We propose a new token model that allows for participating actors to create tokens containing new data, to grant consent to other actors to process data from a token and to link data tokens together to represent data processing flows. As shown in Figure 1, our proposed token model contains:

- Identifier (ID)
- Hash of the data content, used as IPFS address
- A list of IDs of input tokens

User consent is represented in a separate data structure that includes:

- Hash of the token
- Address of the token owner
- Address of the user to consent access to (a data processor)
- Number of accesses allowed

Figure 2 introduces our GDPR-compliant token API which contains the main features of our model:

- **createToken** generates a new token and adds it to the list of tokens. After that, it triggers the *TokenCreated* event.
- **viewToken** returns the token attributes to its owner or to a user who possesses an access consent.
- **viewTokensList** returns the list of tokens owned by a certain user.
- **addDataToToken** adds content to an existent token by its owner and triggers the *DataAdded* event.
- **addOutputToToken** finalizes a token represented by its *_tokenID* and adds *_outputAddress* as an output of the token by generating a new consent containing the *_tokenID*, the token owner address, the *_outputAddress*, the token's hash and *_access* which represents the number of times the output can access the token. Then it generates the *OutputAdded* event. Once finalized, a token can no longer be modified.
- **addInputToToken** verifies if a consent including *_inputTokenID* and the user address exists and checks if $\_access > 0$. If these conditions are met, the function adds the token with *_inputTokenID* identifier as an input

to the token with *_tokenID* identifier. Then it triggers the *InputAdded* event.
- **deleteToken** makes the token "unavailable" and triggers the *TokenDeleted* event.

```
interface PrivacyCompliantToken{

//Events
event TokenCreated(uint256 ID);

event DataAdded(uint256 ID, string _data);

event OutputAdded(uint256 consentID, uint256 tokenID,
     address owner, address output);

event InputAdded(uint256 ID, address owner, int8 times);

event TokenDeleted(uint256 ID);


//Functions
function createToken(string calldata _data) external;

function viewToken(uint256 _ID, int256 _consentID) public
     view onlyAuthorized (int256(_ID), _consentID, 1)
     returns (uint256 ID, string memory _data, uint256[]
     memory _inputID, bool _available);

function viewTokensList() public view returns (uint256[]
     memory);

function addDataToToken(string calldata _data, uint256 _ID)
      external onlyOwner(_ID);

function addOutputToToken(uint256 _tokenID, address
     _outputAddress, int8 _access) external onlyOwner(
     _tokenID);

function addInputToToken(uint256 _tokenID, uint256
     _inputTokenID) external onlyOwner(_tokenID);

function deleteToken(uint256 _ID) external onlyOwner(_ID);

}
```

Fig. 2. Privacy-Compliant Token API

B. Token Operations

We show how our model works using a simple scenario: *Alice* adds file containing her personal data to the system and *Bob* will process this data to generate results. To do so, *Alice* needs to create a token representing her personal data. *Bob* must also create his own token to host the processing result and link it to Alice's token to keep track of the processing flow. This is only possible if *Bob* obtains the consent of *Alice* to access her token. This scheme makes it possible to model not only the data collection, but also the processing of this data while keeping track of the processing result.

As a primary step, *Bob* needs to provide his public key to *Alice* in order to encrypt the personal data. Figure 3 shows the different steps taken:

- *Token Creation: Alice* uploads her encrypted data to IPFS (1a), pins it in order to ensure its persistence and gets its hash in return (1b). Next, she calls *createToken* to create a new token with the hash of the data (1c). *Bob* follows the same steps in (2a) and (2b) to create and empty token.
- *Output Addition: Alice* adds an output to her token by calling *addOutputToToken* (3). This function finalizes the token (i.e. the token's attributes become non-modifiable) and creates a consent which authorizes Bob to use this token from Alice.
- *Input Addition: Bob* adds an input to his token by calling *addInputToToken* (4) with Alice's token as input. This function verifies that the specified token contains an active consent to the caller address. Successfully completing this step allows Bob to demonstrate that the results contained in his token were generated using data from Alice, and that he has obtained her consent to use the data in her token.
- *Token View:* To view Alice's personal data, *Bob* calls *tokenView* function (5a) from which, he can extract the data hash (5b). Next, he gets the encrypted data from IPFS (5c) (5d). Finally, he decrypts the data using his private key.
- *Processing Result Addition:* To add the result of the processing, *Bob* needs to follow the steps in (6a), (6b), and (6c) to add the result's hash to his token.
- *Token Deletion:* To delete her token, *Alice* calls *tokenDelete* function (6) makes the token unavailable for reading. Then, she unpins the data from the IPFS in order to be removed in the next garbage collection phase.

C. Architecture Components

Figure 4 shows our reference system architecture. In addition to the Ethereum blockchain, and the integration with IPFS, the architecture contains several components for the web client application:

- **Web3.js:** It is a set of libraries that provides an Ethereum JavaScript API that enables the user to interact with the Ethereum network and manage Ethereum accounts.
- **MetaMask:** It acts as a bridge between the Ethereum network and the web application by injecting Web3 to the application and by hosting the Ethereum wallet in the client browser.

Based on this configuration, when a user wants to execute a smart contract function, Web3 converts this request into an Ethereum transaction which will be signed and sent by MetaMask. These transactions interact with the *PrivacyCompliantToken* contract which is written in Solidity, compiled and deployed on Ethereum.

In addition, the web application includes **Ipfs-http-client.js** which is a Javascript API that enables to interact with an IPFS node which allows the user to execute all of the IPFS functions including file addition, retrieval, and deletion.

V. Analysis of GDPR Compliance

In this section, we explain how our proposed data tokenization model and reference architecture provides support for GDPR. We furthermore demonstrate GDPR compliance through an example scenario for machine learning training.

A. Support for GDPR Rights

Our proposed solution supports GDPR regulations as follows:

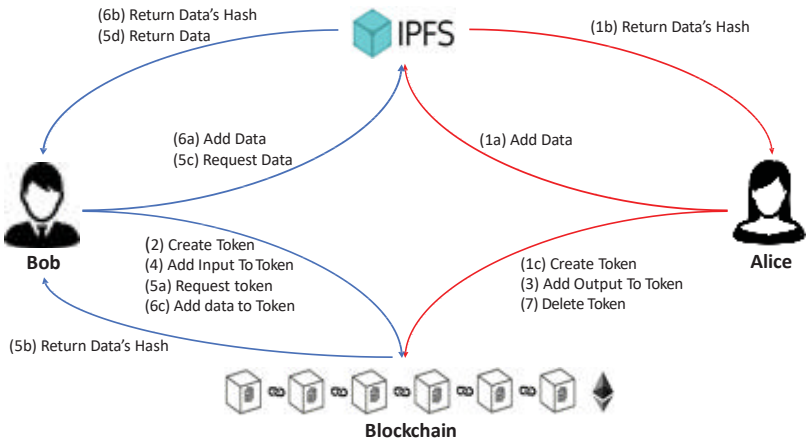

Fig. 3. System Interaction Overview

- **The right to restrict and control data access:** This right is granted through the *Consent* data structure which controls who has the right to access the personal data and limits the number of uses (Access Times).
- **The right to delete the personal data:** Any data collected must be uploaded and pinned to the off-chain storage layer (*IPFS*). Erasure is provided by our deletion mechanism that consists on unpinning the content from IPFS and making the on-chain token unavailable. Once unpinned, the content will be garbage collected and be unavailable for downloading. The only thing that will remain afterwards is a hash of the content on-chain, which is necessary for auditability purpose. We argue that maintaining the hash on-chain does not violate our interpretation of the GDPR since the original content cannot be reconstructed. Furthermore, unpinned content typically disappear from IPFS within 24 hours [23], which is below the 1 month threshold required by the GDPR. However, a malicious actor could intentionally keep a certain file available even when the pinning rewards have disappeared. Our interpretation therefore argues that a probabilistic guarantee that the data will be erased is sufficient to satisfy the GDPR due to the lack of economic incentives to keep the data available. More explicit content deletion mechanisms for GDPR, such as blacklisting[2], will be explored in future works. Note that we also interpret the right of erasure to only affect the source data from the users, and not require deletion of results involving this data. This latter is also left as future work.
- **The right of transfer:** Using our system, a data owner can request its data at any time. To do so, the user can view the content of its own tokens, which is verified by the smart contract. Then, the user retrieves the data files from IPFS using the hashes from the tokens.
- **Tracking the audit trail of personal data usage:** We consider the blockchain as the authoritative record of data processing activities by the data processors. Any results generated from the data processors should be captured in data tokens, with the appropriate inputs referring to tokens containing the source data used to generate those results. During an audit process, a company may be caught and fined for collecting data which has not be traced on chain, producing results which are inconsistent with the hashes of the recorded results, incorrectly listing which input data was used to produce a result token, or continuing to store data for deleted tokens. Note that a data processor could also source its own results to a third company, thus creating longer chains of linked data tokens, which records end-to-end provenance.

B. Use Case Study: Machine Learning Training

In order to give a concrete usage of our system and to demonstrate how GDPR compliance is verified, we describe a mobile service use case: in order to provide adaptive Quality of Service and to study proper customer mobile plans, a mobile service provider uses its cellular network to collect the client data (age, gender, geographical position...) in order to train two machine learning models that track mobile service usage patterns and provides insights for the aforementioned purposes.

The clients upload their personal information files to IPFS, pin them and create tokens containing the hash of the files. The mobile service provider creates a token for each model and then adds the results of training the two models.

To grant access to its data, a client needs to create a consent with the mobile service provider address as an output. The client has two options:

- **Option 1:** The client can provide one access time. This will allow the mobile service provider to use the client data for one of its machine learning models and prevents it from using it for both of them.

[2]https://github.com/decentraland/ipfs-node/issues/14

- **Option 2:** The client can provide two access times. This will allow the mobile service provider to use the client data for both models.

The final step of the authorization is to add the client tokens as input to the provider's model tokens. This way, the mobile service provider will be authorized to use the personal data to train the models with respect to GDPR regulations. A client can always stop the access to the personal data files by triggering the deletion process.

In order to verify the integrity of the machine learning model and to check whether the input data was subject to a prior consent or not, the auditor can follow this approach:

- Gather the input data from the machine learning model token,
- Train the model with the collected input data,
- Compare the training results with the existing results,
- If we obtain equal results, we can conclude that the model was trained with the exact data collected with consent. Otherwise, the model was trained with an illegally obtained data.

To confirm the respect of the proper operation sequencing to obtain the access to clients' personal data, the auditor can follow this approach:

- Select an input token from the machine learning model token,
- Get the height of the blocks that contain the transactions which create the output for the client data token and the transactions which create the input for the machine learning model token,
- In the normal case, the output creation (consent creation) block height should be inferior the input creation block height.
- Repeat the previous operations with the remaining input tokens.

To validate the deletion of a token, the auditor requests the data IPFS hash from the deleted token which is kept for this purpose. Then, it tries to request the data from the IPFS. If the data was actually deleted, the auditor should not receive a response from the network.

Finally, to validate the right to transfer, the auditor can request the hash of some data from its token and use the hash to retrieve the data from the IPFS. The auditor should be able to obtain a copy of the data.

VI. Evaluation

To evaluate our *Privacy-Compliant Data Collection System*, we create different synthetic scenarios. These tests aim to evaluate the system performance in terms of gas consumption and overall latency.

We use the following evaluation setup: the scenarios are scripted in NodeJS and executed on a laptop equipped with Intel(R) Core(TM) i7 1.90GHz processor running on Windows 10. The smart contract is deployed on a Ganache private Ethereum network. We decide to use 16 gwei as the gas price which is the average at the time of the experiment according to *Etherscan* with an average confirmation time of 37 seconds. In each scenario, 100 operation are executed by 3 users.

A. Token Creation Rate Effect

In this experiment, we evaluate the effect of token creation rate on the total gas consumption. We run the system 8 times with a fixed number of operations (100 operation) including token creation operations alongside other operations. In each execution, we increase the rate of token creation in comparison with other operations. As shown in Figure 6, the increase of token creations rises linearly with the overall gas consumption. We measure the gas consumption of each operation and find that *createToken* costs 276476 gas unit (Figure 5), which is higher than the average gas consumption of the other operations of 102355 gas unit. This is due to the cost of storing a new data structure on-chain along with the complexity of the *createToken* function in terms of operations number.

B. Output Rate per Token Effect

In this experiment, we evaluate the effect of increasing the number of outputs per token on the overall gas consumption. We run the system 4 times with 100 operation including 20 *createToken*. For each iteration, we increase the number of output per token and measure the overall gas consumption of each execution. As shown in Figure 7, the increase in the number of output per token is followed by a significant increase in the gas consumption. This is due to the cost of *addOutput* which rises to 196157 gas unit (Figure 5). The reason of this gas consumption is the creation of a new consent data structure to the smart contract storage.

C. Input Rate per Token Effect

In this experiment, we evaluate the impact of increasing the input addition per token on the overall gas consumption. We launch the test 4 times with 100 operation including 20 *createToken*. For each iteration, we increase the number of input per token and measure the total gas consumption. Figure 8 shows a decrease in the gas consumption which is due to the cost of *addInput* function which is equal to 81052 gas unit. This is due to the simplicity of *addInput*'s instructions.

D. Data Size Effect

The purpose of this experiment is to determine the impact of the data size on the system latency. We generate different dummy files with different sizes and for each size we run the experiment using the same configuration: we generate 100 operations including 18 *createToken* and 18 *viewToken*. Given that Ganache do not consider the transaction confirmation delay, we define a fixed delay which is equals to 37 seconds. This value corresponds to the average confirmation delay for a gas price equals to 16 gwei at the time of the experiment according to *Etherscan Gas Tracker*. As shown in Figure 9, the overall system latency is almost constant (~62.47min). This proves that the data size does not affect the system latency.

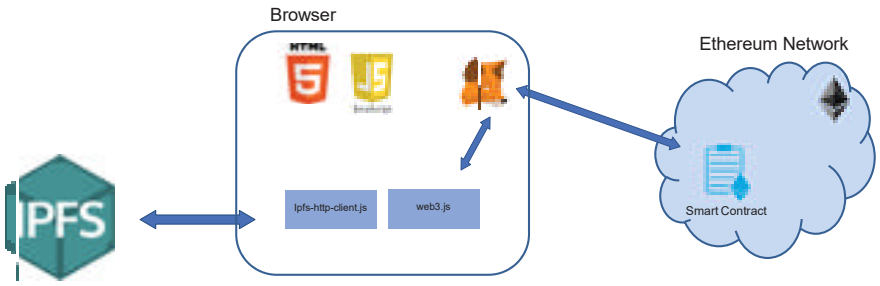

Fig. 4. System Architecture

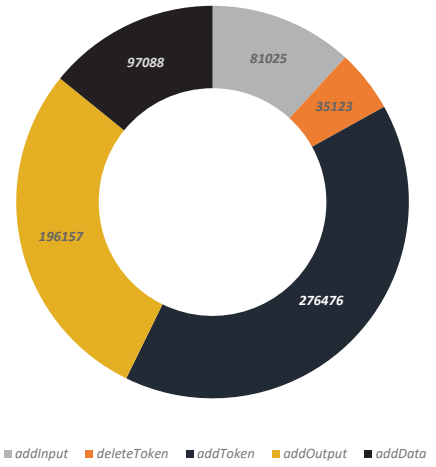

Fig. 5. Gas Consumption Breakdown

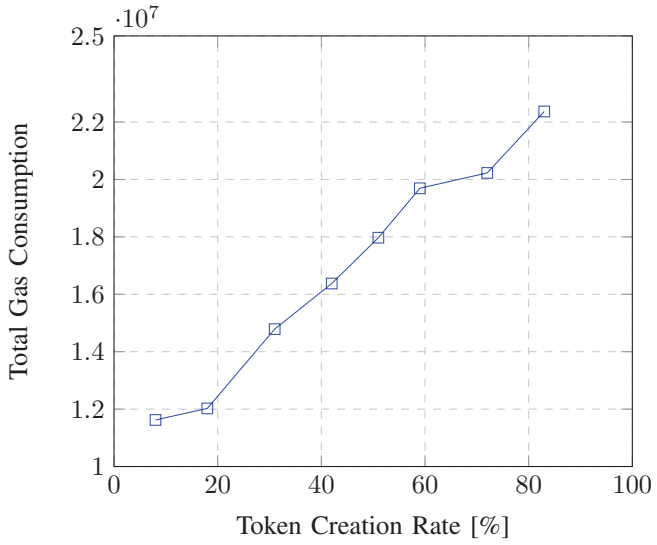

Fig. 6. Total Gas Consumption per Token Creation Rate

E. IPFS Cost Efficiency

To demonstrate the cost efficiency of the IPFS in comparison with the on-chain storage, we investigate the cost of storing data on a smart contract storage which can be considered as a large array of "words" of 32 bytes. According to [24], **SSTORE** is the opcode responsible for storing a word on the storage and costs 20000 gas. With the considered gas cost of 16 gwei and the conversion rate of 165.52 usd for 1 eth, we calculate the cost of storing different amounts of data on-chain. The collected results presented in Table I shows that the cost of storing a small amount of data on the smart contract storage is very high (e.g., storing 100 kbytes costs 165.52 usd). On the other hand, using an IPFS pinning service is much cheaper. For example, "Pinata" offers a free service when the amount of data is under 1 Gbyte and 0.15 usd/month fees for data above 1 Gbyte. Although storing on-chain requires a one-time payment, it is still cheaper and more convenient to store data on IPFS.

TABLE I
ON-CHAIN STORAGE COST

Data Size (kbytes)	Storing Cost (gas)	Storing Cost (ETH)	Storing Cost (USD)
1	640000	0.01	1.69
10	6260000	0.1	16.57
100	62500000	1	165.52
1000	625000000	10	1655.20

F. Discussion

We observe that the *createToken* and *addOutput* functions consumes significant amount of gas. With the same gas cost and conversion rate used for the experiments, the transaction fee in gwei for the creation and the output addition functions are respectively 4423616 (0.73 USD) and 3138512 (0.52 USD). According to Etherscan, the average transaction fee is ~0.31 usd. Note that the fees can be decreased by reducing the gas price in exchange for increased confirmation time. For example, if we define a 8 gwei/gas, the fees will be 0.37 usd and the confirmation time could go to 3 minutes.

According to the results, we argue that an optimized usage pattern of our system uses a limited quantity of tokens, low quantity of output creation, but a higher number of inputs per token. Our system scales well with large file sizes. The example described in Section V is therefore an appropriate use

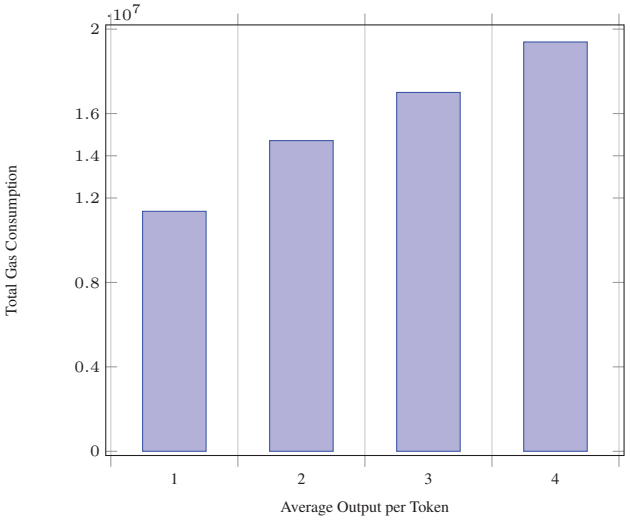

Fig. 7. Total Gas Consumption per Average Output per Token

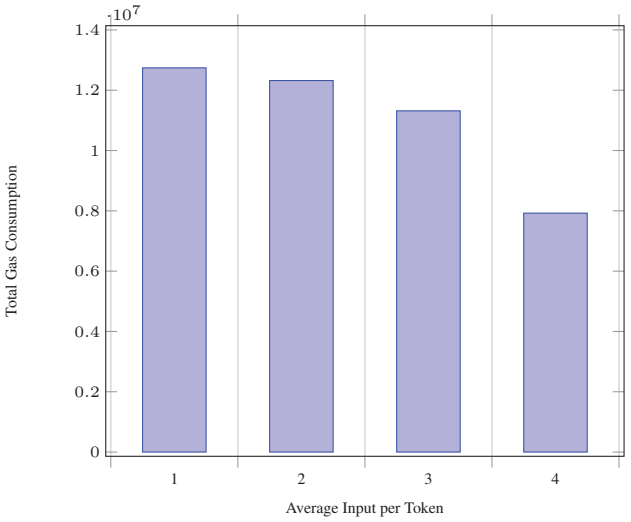

Fig. 8. Total Gas Consumption per Average Input per Token

case from a performance standpoint. For n clients, 1 token and 2 outputs will be created by each client, and 2 tokens and $2 \times n$ inputs will be created by the mobile service provider.

VII. Conclusion

With the enforcement of the new GDPR, organizations need to employ new mechanisms to comply with its privacy regulations. We propose an innovative combination of a decentralized file storage system such as IPFS and a blockchain DLT to build a privacy-compliant data collection system. Specifically, we introduce a new data tokenization model that enables data owners to define consent policies and control the access. We use Ethereum as an immutable ledger which records audit trails of data processing activities, and as a smart contract platform to validate data operations with GDPR compliance. IPFS provides tunable availability, which can be leveraged to support the right of erasure.

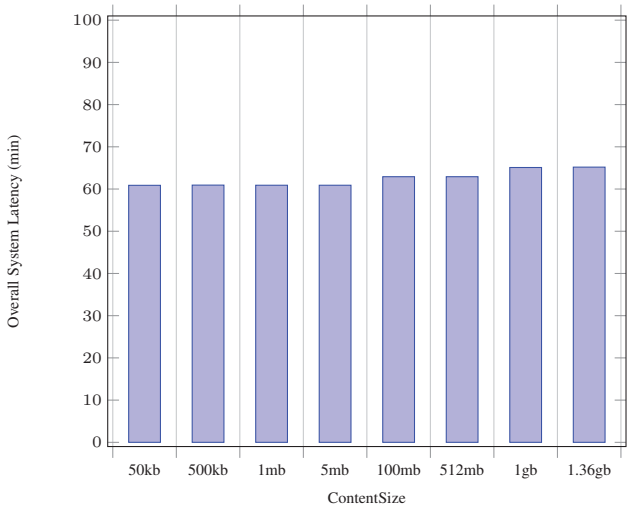

Fig. 9. Impact of Content Size on System Latency

We implemented our system using a smart contract written in Solidity. The results show that the token creation and the consent creation functions have a significantly higher gas consumption than the other functions. Furthermore, the data size has no effect on the system latency. We argue that a gas price adjustment is a possible solution to the gas cost. In addition, we observe that the studied use case of machine learning training matches well with the optimal usage pattern for performance.

References

[1] B. Parno, J. M. McCune, D. Wendlandt, D. G. Andersen, and A. Perrig, "Clamp: Practical prevention of large-scale data leaks," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 154–169.

[2] M. Goddard, "The eu general data protection regulation (gdpr): European regulation that has a global impact," *International Journal of Market Research*, vol. 59, no. 6, pp. 703–705, 2017.

[3] K. Zhang and H.-A. Jacobsen, "Towards dependable, scalable, and pervasive distributed ledgers with blockchains," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1337–1346.

[4] V. Buterin and F. Vogelsteller, "Erc20 token standard," *URL: https://theethereum. wiki/w/index. php/ERC20 Token Standard*, 2015.

[5] W. Entriken, D. Shirley, J. Evans, and N. Sachs, "Erc-721 non-fungible token standard," *Ethereum Foundation*, 2018.

[6] A. Ouaddah, A. A. Elkalam, and A. A. Ouahman, "Towards a novel privacy-preserving access control model based on blockchain technology in iot," in *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Springer, 2017, pp. 523–533.

[7] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "Creditcoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2204–2220, 2018.

[8] R. Yuan, Y.-B. Xia, H.-B. Chen, B.-Y. Zang, and J. Xie, "Shadoweth: Private smart contract on public blockchain," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 542–556, 2018.

[9] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 839–858.

[10] N. Kshetri, "Blockchain's roles in strengthening cybersecurity and protecting privacy," *Telecommunications policy*, vol. 41, no. 10, pp. 1027–1038, 2017.

[11] Z. Wu, A. B. Williams, and D. Perouli, "Dependable public ledger for policy compliance, a blockchain based approach," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1891–1900.

[12] C.-K. Chu, S. S. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 2, pp. 468–477, 2013.

[13] S. Sundareswaran, A. Squicciarini, and D. Lin, "Ensuring distributed accountability for data sharing in the cloud," *IEEE transactions on dependable and secure computing*, vol. 9, no. 4, pp. 556–568, 2012.

[14] X. Liu, Y. Zhang, B. Wang, and J. Yan, "Mona: Secure multi-owner data sharing for dynamic groups in the cloud," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 6, pp. 1182–1191, 2012.

[15] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, "Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability," in *Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing*. IEEE Press, 2017, pp. 468–477.

[16] Z. Li, A. V. Barenji, and G. Q. Huang, "Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform," *Robotics and Computer-Integrated Manufacturing*, vol. 54, pp. 133–144, 2018.

[17] P. Cuccuru, "Beyond bitcoin: an early overview on smart contracts," *International Journal of Law and Information Technology*, vol. 25, no. 3, pp. 179–195, 2017.

[18] V. Buterin *et al.*, "Ethereum white paper," *GitHub repository*, pp. 22–23, 2013.

[19] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance evaluation of the quorum blockchain platform," *arXiv preprint arXiv:1809.03421*, 2018.

[20] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[21] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[22] G. D. P. Regulation, "Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46," *Official Journal of the European Union (OJ)*, vol. 59, no. 1-88, p. 294, 2016.

[23] M. Steichen, B. Fiz, R. Norvill, W. Shbair, and R. State, "Blockchain-based, decentralized access control for ipfs," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1499–1506.

[24] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

ANNEXE II

LE CODE DU CONTRAT INTELLIGENT

```
//solium-disable linebreak-style
pragma solidity >=0.5.0;
pragma experimental ABIEncoderV2;



contract NonFungibleTokenV03{

    uint256 internal tokenCounter;
    uint256 internal consentCounter;
    struct token {
        uint256 ID;
        string[] content;
        uint256[] inputID;
        bool available;
        bool finalized;
    }
    struct consent{
        uint256 ID;
        uint256 tokenID;
        bytes32 tokenHash;
        address owner;
        address output;
        int8 access;
        bool active;
    }

    consent[] internal createdConsents;
    token[] internal createdTokens;
    token[] internal deletedTokens;

    mapping (uint256 => address) internal tokenToOwner;
    mapping (address => uint256[]) internal ownerToTokens;
    mapping (address => uint256[]) internal outputToConsentList;
    mapping (uint256 => uint) internal tokenIdToIndex;
    mapping (uint256 => uint) internal tokenIdToDeleted;
    mapping (uint256 => uint) internal consentIdToIndex;

//modifiers
    modifier onlyOwner (uint256 _ID){
```

```
        require(tokenToOwner[_ID] == msg.sender,"Message sender does not own this token ");
        _;
    }

    modifier onlyAuthorized (int256 _ID, int256 _consentID, uint option){
        if (option == 1){
            bool authorized = false;
            if ((_consentID < 0)){
                if((tokenToOwner[uint256(_ID)] == msg.sender))
                    authorized = true;
            }
            else{
                uint consentIndex;
                uint256 consentID = uint256(_consentID);
                consentIndex = consentIdToIndex[consentID];
                consent storage _consent = createdConsents[consentIndex];
                if ((_consent.output == msg.sender)
                    && (_consent.tokenID == uint256(_ID))
                    && (_consent.active)){
                    authorized = true;
                }
            }
            require((authorized), "Message sender does not have access to this token");
            _;
        }
        else if (option == 2){
            bool authorized = false;
            uint index = consentIdToIndex[uint256(_consentID)];
            consent storage _consent = createdConsents[index];
            if ((_consent.output == msg.sender) && (_consent.active)){
                    authorized = true;
                }


            require ((tokenToOwner[uint256(_ID)] == msg.sender) || (authorized), "Message
                sender does not have access to this token");
            _;
        }
        else revert("Message sender does not have access to this token");
    }

/*****
********************
*****Evenements*****
```

```
********************
*****/
    event TokenCreated(uint256 ID);
    event DataAdded(uint256 ID, string _data);
    event OutputAdded(uint256 consentID, uint256 tokenID, address owner, address output);
    event InputAdded(uint256 ID, address owner, int8 times);
    event NotAuthorized(string _message);
    event TokenDeleted(uint256 ID);

/*****
********************
**functions_publics*
********************
*****/
    constructor ()  public {
        tokenCounter = 0;
        consentCounter = 0;
    }


    function idView() public view returns (uint256) {
        return(tokenCounter);
    }

    function tokenView (uint256 _ID, int256 _consentID) public view onlyAuthorized(int256(_ID)
        , _consentID, 1)
    returns (uint256 ID, string memory _data, uint256[] memory _inputID, bool _available){
        uint index = tokenIdToIndex[_ID];
        token memory _token = createdTokens[index];
        if (_token.available == true){
          string memory content = _token.content[0];

          for (uint i = 1; i < _token.content.length; i++)
          content = string(abi.encodePacked(content,"x00000x" ,_token.content[i]));

        return(_token.ID, content, _token.inputID, _token.available);
      }
        else {
            ID = _token.ID;
            _available = _token.available;
        }
    }

    function tableView () public view returns (uint){
```

```
        return (createdTokens.length);
    }

    function tokenstableView () public view returns (uint256[] memory){
        return (ownerToTokens[msg.sender]);
    }


    function consentView (uint256 consentID) public view onlyAuthorized(-1, int256(consentID),
         2)
    returns (uint256 _consentID, uint256 _tokenID, address _owner, address _output, int8
        _access, bool _active ){
        uint index = consentIdToIndex[consentID];
        consent memory _consent = createdConsents[index];
        return(_consent.ID, _consent.tokenID, _consent.owner, _consent.output, _consent.access
            , _consent.active);
    }


    function createToken (string calldata _data) external {
        uint256  tokenID = _createNewToken();
        _addDataToLastToken(_data);
        emit TokenCreated(tokenID);
    }


    function addDataToToken(string calldata _data, uint256 _ID) external onlyOwner(_ID) {
        uint index = tokenIdToIndex[_ID];
        token storage _token = createdTokens[index];
        if(!_token.finalized){
            _token.content.push(_data);
            emit DataAdded(_token.ID, _token.content[_token.content.length - 1]);
        } else {
            revert("Data can not be added to this token. Reason: Finalized");
        }
    }


    function addInputToToken (uint256 _ownID, uint256 _inID) external onlyOwner(_ownID) {
        if (_validateToken(_ownID, _inID)){
            uint index = tokenIdToIndex[_ownID];
            token storage _token = createdTokens[index];
            (bool authorized, int8 _access) = _authorizeAdd(_inID, msg.sender);
            if (authorized){
```

```
                _token.inputID.push(_inID);
                emit InputAdded(_inID, tokenToOwner[_inID], _access);
            }
            else emit NotAuthorized("Action Not Authorized");
        }
        else revert("false");
    }

    function addOutputToToken(uint256 _ownID, address _out, int8 _access) external onlyOwner(
        _ownID) {
        if (msg.sender != _out){
            uint index = tokenIdToIndex[_ownID];
            token storage _token = createdTokens[index];
            _token.finalized = true;
            bytes memory encodedData = abi.encode(_token.ID, _token.content, _token.inputID);
            bytes32 tokenHash = keccak256(encodedData);
            _createNewConsent(_ownID, tokenHash, msg.sender, _out, _access);
            consent storage _consent = createdConsents[createdConsents.length - 1];
            emit OutputAdded(_consent.ID, _consent.tokenID, _consent.owner, _consent.output);
        }
        else revert("false");
    }

    function deleteToken(uint256 _ID) external onlyOwner(_ID){
        token storage _token = createdTokens[tokenIdToIndex[_ID]];
        delete _token.content;
        //toBeChecked
        delete _token.inputID;
        _token.available = false;
        emit TokenDeleted(_ID);
    }

/*****
********************
**fonctions_privees*
********************
*****/

    function _createNewToken() internal returns (uint256) {
        token memory newToken;
        tokenCounter++;
        newToken.ID = tokenCounter;
        newToken.available = true;
        newToken.finalized = false;
```

```
        createdTokens.push(newToken);
        tokenIdToIndex[newToken.ID] = createdTokens.length - 1;
        tokenToOwner[newToken.ID] = msg.sender;
        ownerToTokens[msg.sender].push(newToken.ID);
        return (newToken.ID);
    }

    function _createNewConsent(uint256 _tokenID, bytes32 _tokenHash, address _tokenOwner,
        address _outputAddress, int8 _access) internal {
        consent memory newConsent;
        consentCounter++;
        newConsent.ID = consentCounter;
        newConsent.tokenID = _tokenID;
        newConsent.tokenHash = _tokenHash;
        newConsent.owner = _tokenOwner;
        newConsent.output = _outputAddress;
        newConsent.access = _access;
        newConsent.active = false;
        createdConsents.push(newConsent);
        consentIdToIndex[newConsent.ID] = createdConsents.length - 1;

    }

    function _addDataToLastToken(string memory _data) internal {
        token storage _token = createdTokens[createdTokens.length - 1];
        if(!_token.finalized){
        _token.content.push(_data);
        emit DataAdded(_token.ID, _token.content[_token.content.length - 1]);
        } else {
            revert("Data can not be added to this token. Reason: Finalized");
        }
    }


    function _authorizeAdd(uint256 _tokenID, address _output)  internal returns (bool, int8) {
        bool _addable = false;
        uint256 consentID;
        uint consentIndex;
        int8 _times;
        for (uint i = 0; i < createdConsents.length; i++){
            consent storage _consent = createdConsents[i];
            if ((_consent.output == _output) && (_consent.tokenID == _tokenID)){
                if ((!_consent.active) && (_consent.access > 0)){
                    _consent.active = true;
```

```
                    _addable = true;
                }
                _times= _consent.access;
                _consent.access--;
                if(_consent.access == 0)
                    _consent.active = false;
            }

        }
        return (_addable, _times);
    }

    function _validateToken(uint256 _ownID, uint256 _outID) internal view returns (bool){
        if ((tokenToOwner[_outID]!=address(0)) && (tokenToOwner[_ownID]==msg.sender) && (
            tokenToOwner[_ownID] != tokenToOwner[_outID]))
            return true;
        return false;
    }

}
```

RÉFÉRENCES

Antonopoulos, A. M. & Wood, G. (2018). *Mastering ethereum : building smart contracts and dapps*. O'reilly Media.

Badr, B., Horrocks, R. & Wu, X. B. (2018). *Blockchain By Example : A developer's guide to creating decentralized applications using Bitcoin, Ethereum, and Hyperledger*. Packt Publishing Ltd.

Baliga, A., Subhod, I., Kamat, P. & Chatterjee, S. (2018). Performance evaluation of the quorum blockchain platform. *arXiv preprint arXiv :1809.03421*.

Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv :1407.3561*.

Bogner, A., Chanson, M. & Meeuw, A. (2016). A decentralised sharing app running a smart contract on the ethereum blockchain. *Proceedings of the 6th International Conference on the Internet of Things*, pp. 177–178.

Buterin, V. & Vogelsteller, F. (2015). ERC20 Token Standard. *URL : https ://theethereum. wiki/w/index. php/ERC20 Token Standard*.

Buterin, V. et al. (2013). Ethereum white paper. *GitHub repository*, 22–23.

Chu, C.-K., Chow, S. S., Tzeng, W.-G., Zhou, J. & Deng, R. H. (2013). Key-aggregate cryptosystem for scalable data sharing in cloud storage. *IEEE transactions on parallel and distributed systems*, 25(2), 468–477.

Cuccuru, P. (2017). Beyond bitcoin : an early overview on smart contracts. *International Journal of Law and Information Technology*, 25(3), 179–195.

Entriken, W., Shirley, D., Evans, J. & Sachs, N. (2018). Erc-721 non-fungible token standard. *Ethereum Foundation*.

Goddard, M. (2017). The EU General Data Protection Regulation (GDPR) : European regulation that has a global impact. *International Journal of Market Research*, 59(6), 703–705.

Iansiti, M. & Lakhani, K. R. (2017). The truth about blockchain. *Harvard Business Review*, 95(1), 118–127.

Kosba, A., Miller, A., Shi, E., Wen, Z. & Papamanthou, C. (2016). Hawk : The blockchain model of cryptography and privacy-preserving smart contracts. *2016 IEEE symposium on security and privacy (SP)*, pp. 839–858.

Kshetri, N. (2017). Blockchain's roles in strengthening cybersecurity and protecting privacy. *Telecommunications policy*, 41(10), 1027–1038.

Li, L., Liu, J., Cheng, L., Qiu, S., Wang, W., Zhang, X. & Zhang, Z. (2018a). Creditcoin : A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(7), 2204–2220.

Li, Z., Barenji, A. V. & Huang, G. Q. (2018b). Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform. *Robotics and Computer-Integrated Manufacturing*, 54, 133–144.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K. & Njilla, L. (2017). Provchain : A blockchain-based data provenance architecture in cloud environment with enhanced

privacy and availability. *Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing*, pp. 468–477.

Liu, X., Zhang, Y., Wang, B. & Yan, J. (2012). Mona : Secure multi-owner data sharing for dynamic groups in the cloud. *IEEE transactions on parallel and distributed systems*, 24(6), 1182–1191.

Maymounkov, P. & Mazieres, D. (2002). Kademlia : A peer-to-peer information system based on the xor metric. *International Workshop on Peer-to-Peer Systems*, pp. 53–65.

McCorry, P., Shahandashti, S. F. & Hao, F. (2017). A smart contract for boardroom voting with maximum voter privacy. *International Conference on Financial Cryptography and Data Security*, pp. 357–375.

Mercer, R. (2016). Privacy on the blockchain : Unique ring signatures. *arXiv preprint arXiv :1612.01188*.

Nakamoto, S. et al. (2008). Bitcoin : A peer-to-peer electronic cash system.

Ouaddah, A., Abou Elkalam, A. & Ait Ouahman, A. (2016). FairAccess : a new Blockchain-based access control framework for the Internet of Things. *Security and Communication Networks*, 9(18), 5943–5964.

Ouaddah, A., Elkalam, A. A. & Ouahman, A. A. (2017). Towards a novel privacy-preserving access control model based on blockchain technology in IoT. Dans *Europe and MENA Cooperation Advances in Information and Communication Technologies* (pp. 523–533). Springer.

Parno, B., McCune, J. M., Wendlandt, D., Andersen, D. G. & Perrig, A. (2009). CLAMP : Practical prevention of large-scale data leaks. *2009 30th IEEE Symposium on Security and Privacy*, pp. 154–169.

Ramsay, S. (2018). The General Data ProtectionRegulation vs. The Blockchain : A legal study on the compatibility between blockchain technology and the GDPR.

Regulation, G. D. P. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46. *Official Journal of the European Union (OJ)*, 59(1-88), 294.

Sel, D., Zhang, K. & Jacobsen, H.-A. (2018). Towards Solving the Data Availability Problem for Sharded Ethereum. *Proceedings of the 2Nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, (SERIAL'18), 25–30. doi : 10.1145/3284764.3284769.

Shahsavari, Y., Zhang, K. & Talhi, C. (2019, April). Performance Modeling and Analysis of the Bitcoin Inventory Protocol. *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, pp. 79-88. doi : 10.1109/DAPPCON.2019.00019.

Steichen, M., Fiz, B., Norvill, R., Shbair, W. & State, R. (2018). Blockchain-Based, Decentralized Access Control for IPFS. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1499–1506.

Sundareswaran, S., Squicciarini, A. & Lin, D. (2012). Ensuring distributed accountability for data sharing in the cloud. *IEEE transactions on dependable and secure computing*, 9(4), 556–568.

Truong, N. B., Sun, K., Lee, G. M. & Guo, Y. (2019). GDPR-compliant personal data management : A blockchain-based solution. *arXiv preprint arXiv :1904.03038*.

van Geelkerken, F. & Konings, K. (2017). Using Blockchain to strengthen the rights granted through the GDPR. *Litteris et Artibus : matériaux*, pp. 458–461.

Wood, G. et al. (2014). Ethereum : A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014), 1–32.

Wu, Z., Williams, A. B. & Perouli, D. (2019). Dependable Public Ledger for Policy Compliance, a Blockchain Based Approach. *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1891–1900.

Xu, L., Shah, N., Chen, L., Diallo, N., Gao, Z., Lu, Y. & Shi, W. (2017). Enabling the sharing economy : Privacy respecting contract based on public blockchain. *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, pp. 15–21.

Yuan, R., Xia, Y.-B., Chen, H.-B., Zang, B.-Y. & Xie, J. (2018). Shadoweth : Private smart contract on public blockchain. *Journal of Computer Science and Technology*, 33(3), 542–556.

Zhang, K. & Jacobsen, H.-A. (2018). Towards Dependable, Scalable, and Pervasive Distributed Ledgers with Blockchains. *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1337–1346.

Zyskind, G., Nathan, O. et al. (2015). Decentralizing privacy : Using blockchain to protect personal data. *2015 IEEE Security and Privacy Workshops*, pp. 180–184.