République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Belhadj Bouchaib d'Aïn-Témouchent

Faculté des Sciences et de Technologie

Département de Mathématiques et de l'Informatique

Mémoire de Fin d'Études

Présenté en vue de l'obtention du Diplôme de Master en Informatique
Spécialité : Réseaux et Ingénierie des Données (RID)

Présenté par :
Mme.Kenza Zeriguine & M.Fayçal Abdelhadi Khaldi

Implémentation d'un protocole d'authentification et d'identification basée sur une preuve à divulgation nulle de connaissance.

Encadrant :
M. Hichem Bouchakour Errahmani
Maître de Conférence "A" , Université Ain Temouchent.

Président du jury :
M. Zoheir Bouafia
Maître Assistant "A" , Université Ain Temouchent.

Examinatrice :
Mme Fatima Zohra Berrakam
Maître Assistant "A" , Université Ain Temouchent.

Promotion 2022 - 2023

Remerciement

الحمد الله الذي هدانا لهذا و ماكنا لنهتدي لولا أن هدانا االله

Tout d'abord nous rendons grâce à **Dieu**,lui qui nous a donné la volonté, le courage, la santé et la patience du travail depuis le début.

Nous exprimons nos profondes gratitudes et respectueuses reconnaissances à notre encadrant *M. Hichem Bouchakour Errahmani* Pour sa bonne volonté d'avoir accepter de nous encadrer et guider grâce ses orientations, ses conseils et ses critiques tout au long de ce travail de recherche en nous laissant la liberté dont on avions besoin. On ne peut que lui être reconnaissant, surtout pour ses qualités intellectuelles et humaines.

Nos remerciements vont aussi au membre du jury, pour l'honneur qu'ils nous ont fait en acceptant d'évaluer ce travail et de participer à la soutenance.

Nous sommes reconnaissants envers tous les enseignants de l'Université Belhadj Bouchaib d'Ain Temouchent pour leurs contributions à notre formation et leurs compétences dans la poursuite de nos études.

Nous remercions nos parents qui nous ont soutenus pendant toute la vie et qui continuerons à nous aider dans tous nos projets à l'avenir. Ainsi que nos frères et sœurs qui ont participé de prés ou de loin et nous ont encouragés et aidés dans notre vie.

Et enfin, nous tenons également à remercier toutes les personnes qui nous ont apportés de l'aide, soit par leurs connaissances dans des domaines spécifiques, soit sous forme de conseils lorsque nous en avions besoin.

Résumé

Ce travail fait partie du projet de fin d'études pour obtenir le diplôme de Master en Réseaux et Ingénierie des Données (*RID*).

Nous souhaitons à travers ce projet étudier , modéliser et implémenter un système d'accréditation et d'authentification dans un contexte décentralisé basé sur les **Zero Knowledge Proofs**. Dans les systèmes décentralisés, où les données des utilisateurs ne sont pas stockées sur un serveur centralisé, il est crucial de garantir la sécurité et l'intégrité des informations. Cependant, la vérification de l'identité et de l'authenticité des données restent un défi majeur. C'est pourquoi nous avons choisi de proposer une méthode d'authentification et d'accréditation basée sur les Zero Knowledge Proofs.

Notre approche consiste à utiliser des protocoles cryptographiques avancés pour permettre la vérification des informations sans révéler des détails confidentiels, ni sur l'accréditation , ni sur son détenteur. Ainsi, les utilisateurs peuvent prouver qu'ils possèdent des informations valides sans avoir à les divulguer explicitement, tout en restant anonyme, afin d'éviter tout suivi ou refus en rapport avec leurs identité.

En mettant en œuvre notre système d'identifiants vérifiables, nous visons à renforcer l'utilité des justificatifs d'identité sur le web. En utilisant les Zero Knowledge Proofs,nous permettont a des utilisateurs de faire vérifier leurs diplomes universitaires par des employeurs potentiels, le tout en assurant l'anonymisation et l'authentification sécurisée.

Pour évaluer les performances de notre système, nous effectuons des tests approfondis et une analyse de sécurité. Les résultats montrent que notre approche offre une efficacité relative en termes de rapidité et garantit une confidentialité des informations d'identification vérifiable.

Mots clés : Mots clés :Authentification, Zero Knowledge Proof, Identité Décentralisé

Abstract

This work is part of the graduation project to obtain the Master's degree in Networks and Data Engineering (*RID*).

We wish through this project to study, model and implement an accreditation and authentication system in a decentralized context based on **Zero Knowledge Proofs**. In decentralized systems, where user data is not stored on a centralized server, it is crucial to ensure information security and integrity. However, verifying the identity and authenticity of data remains a major challenge. This is why we have chosen to offer an authentication and accreditation method based on Zero Knowledge Proofs.

Our approach consists in using advanced cryptographic protocols to allow the verification of information without revealing confidential details, neither on the accreditation, nor on its holder. Thus, users can prove that they have valid information without having to disclose it explicitly, while remaining anonymous, in order to avoid any follow-up or refusal related to their identity.

By implementing our verifiable identifiers system, we aim to make credentials more useful on the web.By using Zero Knowledge Proofs, we allow users to have their university degrees verified by potential employers, while ensuring anonymization and secure authentication.

To assess the performance of our system, we perform extensive testing and security analysis. The results show that our approach provides relative efficiency in terms of speed and guarantees verifiable credential privacy.
Keywords: Authentification , Zero Knowledge Proofs, Decentralised Identity

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Chapitre 1

Introduction générale

Sommaire

1.1 Introduction

La cryptographie a pour but de protéger les données grâce a diverses méthodes en les rendant incompréhensibles pour des personnes non autorisées afin de protéger les informations sensibles contre les interceptions et les modifications .Elle est utilisée dans de nombreux domaines, tels que les communications, les opération militaires, et dans un contexte plus important encore ; l'identification [1] . Le preuve a divulgation nulle de connaissance ou Zero Knowledege Proof (ZKP) est une technique exceptionnelle dans sa manière de sécuriser les données ; en ne les envoyant jamais.

1.2 Motivation

Imaginons que vous êtes en possession de la réponse a une énigme ; Peut-on convaincre que l'on connaît la solution de cette énigme sans dévoiler la réponse ?
Ou bien se connecter a un système sécurisé sans fournir de mot de passe ? Cela peut sembler contradictoire, mais ces opérations sont non seulement possible, mais aussi réalisables avec des algorithmes assez simple au premier abord , le Zero Knowledge Proof, rendant la tâche d'un intrus essayant de prouver une information dont il n'a pas l'accès difficile. [2]

1.3 Problématique

Aujourd'hui la question des données personnelles sur Internet est un sujet majeur de l'écosystème tout entier. Très souvent utilisées à l'encontre des utilisateurs, les données personnelles sont devenues propriété des grandes sociétés. Facebook, Microsoft, Google détiennent aujourd'hui le monopole des données privées [3]. Ils n'hésitent pas à vendre ces informations confidentielles à d'autres sociétés qui vont les utiliser pour mieux cibler leurs publicités par exemple. La solution proposé par le système décentralisé, tel que la blockchain , est de laisser un contrôle total des données aux utilisateurs eux même. C'est ce que on appelle la souveraineté des données, ou Self Sovereign Identity (SSI). Mais une problématique se pose, si chaque personne est souveraine de ces propres données, il faut assurer la vérité totale de ces informations, tout en assurant l'anonymat. Par conséquent, plusieurs travaux ont vu le jour, de sorte où l'on peut se poser les questions suivantes :

- Comment les informations confidentielles peuvent être protégées ?
- Comment prouver une information sans la dévoiler ?
- Comment le ZKP respecte la vie privée ?

C'est en se basant sur l'abondante bibliographie consacrée à la matière, et tout particulièrement sur l'article " The Knowledge complexity of interactive proof-systems" de Goldwasser, Micali et Rackoff, qu'il fut possible de déterminer comment prouver une information avec divulgation nulle de connaissance.

1.4 Organisation du mémoire

Après un chapitre consacré à l'authentification, où nous apprenons les différents protocoles utilisés pour identifier un parti, l'étude du ZKP nous permet de connaître quelques techniques de preuves. Afin de suivre l'évolution des travaux dans le domaine du sujet, nous allons consacrer un chapitre sur l'état de l'art où nous étudions quelques approches. Enfin, nous verrons dans un dernier chapitre notre contribution dans les informations d'identification vérifiables et certificats basé sur le ZKP, en respectant les points étudiés.

Chapitre 2

Authentification et identité numérique

Sommaire

2.1 Introduction

Le principal but de la cryptographie , avant même de sécuriser les données, et d'assurer qu'elle viennent d'une source sure que l'on connaît. Un problème est apparu très rapidement, comment vérifier cette source sans avoir un lien visuel direct ?
C'est **l'authentification**.

L'authentification est apparue en réponse aux besoins de sécurité croissants dans les systèmes informatiques. Les organisations, d'abord militaires et ensuite publiques, ont commencés a utiliser des ordinateurs pour stocker des informations sensibles, et il était nécessaire de mettre en place des mesures de sécurité pour protéger ces données contre les accès non autorisés.

Dans ce chapitre, nous allons d'abord définir ce qu'est l'authentification en présentant ses différents types. Nous entamons par la suite les principales opérations nécessaires à la conception d'un cryptosystème sécurisé, ainsi que les problèmes de conceptions qui surviennent. Enfin, nous aborderons la notion de preuve, qui nous sera de grande utilité pour une bonne compréhension du *Zero Knowledge Proof*

2.2 Authentification

L'authentification est un processus essentiel pour garantir que seuls les personnes autorisées ont accès aux informations sensibles. Il existe plusieurs méthodes d'authentification, notamment la saisie d'un nom d'utilisateur et d'un mot de passe, l'utilisation de cartes à puce ou de jetons d'authentification.
Cependant, toutes ces méthodes ont leurs limites en termes de sécurité. Les noms d'utilisateur et les mots de passe peuvent être piratés, tandis que les cartes à puce et les jetons peuvent être volés ou perdus.

2.2.1 Types d'Authentification

Il existe plusieurs types d'authentification, la connaissance (comme un mot de passe), la possession (comme une carte à puce ou un jeton) et l'identité (comme l'authentification biométrique).

2.3 Les principaux protocoles d'authentification

2.3.1 Authentification basée sur un mot de passe :

Il s'agit de la technique d'authentification d'entité la plus simple et la plus utilisée, en auquel cas le client a une idée du mot de passe. Un mot de passe est utilisé lorsqu'un utilisateur a besoin d'accéder au système. Chaque utilisateur dispose d'une clé d'identification publique et un mot de passe, qui est la clé privée. Ce schéma d'authentification est divisé en 2 types.

Mot de passe fixe :

Un mot de passe fixe est utilisé à plusieurs reprises pour chaque accès.

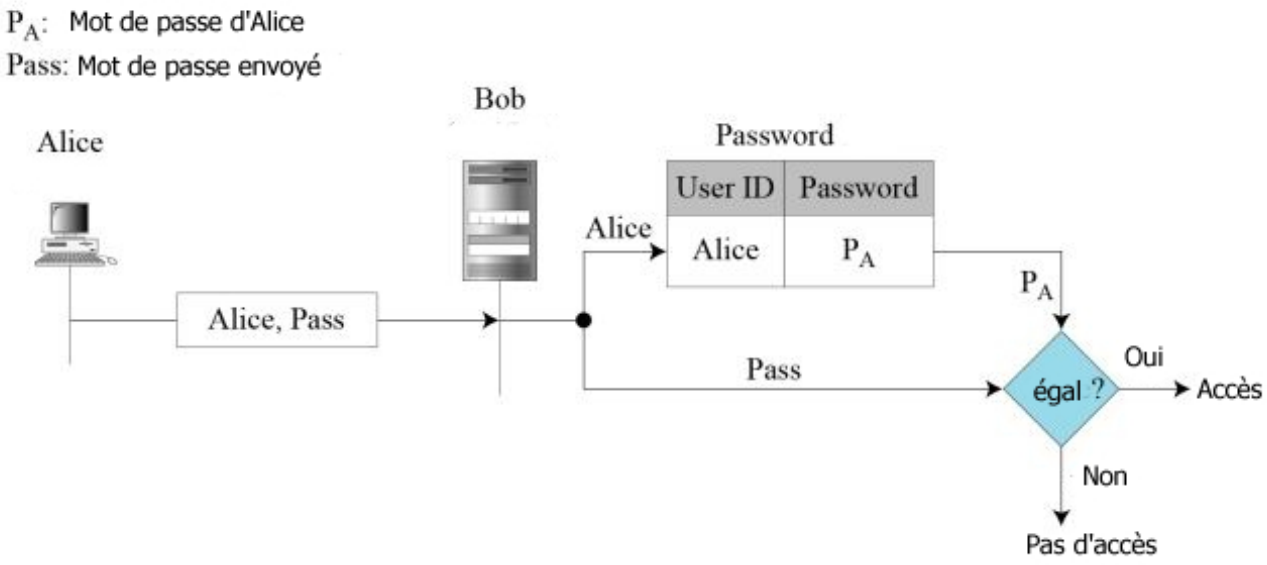


FIG. 2.1 : Authentification avec mot de passe fixe

Mot de passe à usage unique

Il n'est utilisé qu'une seule fois pour obtenir l'accès. Cette sorte de mot de passe rend l'écoute clandestine et le salage inutiles, mais reste très dur a mettre en place.

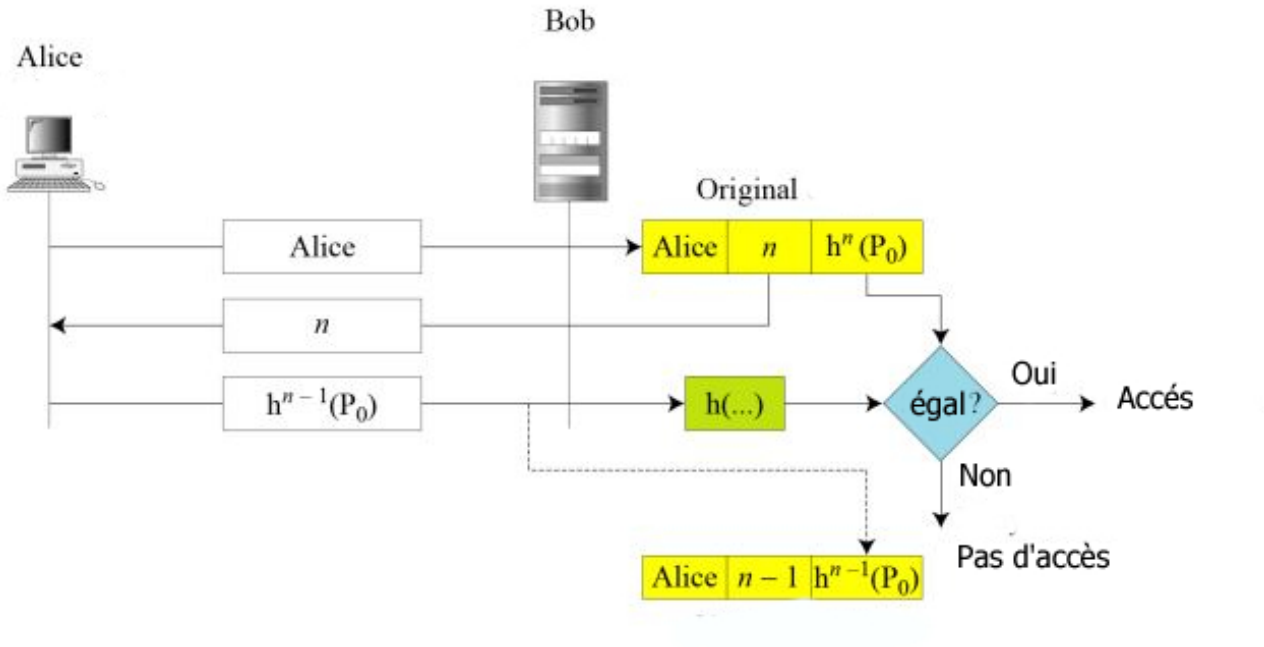


FIG. 2.2 : Authentification avec OTP

2.3.2 Authentification basé sur le défi

Dans l'authentification par mot de passe, le client démontre pour prouver un secret. Cependant, parce que le client révèle ce secret, il s'expose à diverses attaques. Dans l'authentification par défi, le client doit prouver qu'il a connaissance du mot de passe sans en informer le vérificateur. Autrement dit le client n'envoie pas le secret au vérificateur, ce dernier peut le rechercher et le trouver.

Le défi est fondamentalement une valeur variant dans le temps comme n'importe quel nombre aléatoire. Le client applique une fonction au défi et envoie le résultat, qui est appelé comme une réponse, au vérificateur. La réponse montre que le client connaît le secret. Cela peut être fait par quatre méthodes.

- Utilisation d'un chiffrement à clé symétrique
- Utilisation des fonctions de hachage à clé
- Utilisation du chiffrement à clé asymétrique
- Utilisation des signatures numériques

Utilisation d'un chiffrement à clé symétrique

Dans ce type d'authentification, un client et un vérificateur partagent une clé secrète commune qui sert à chiffrer les défis.

Lorsque le vérificateur souhaite mettre le client au défi, il envoie un nonce, c'est-à-dire un nombre aléatoire utilisé uniquement une fois. Ce nonce est crypté en utilisant l'algorithme de chiffrement et la clé secrète partagée. Le client reçoit ensuite le défi chiffré. Le client doit déchiffrer le défi à l'aide de la même clé secrète.

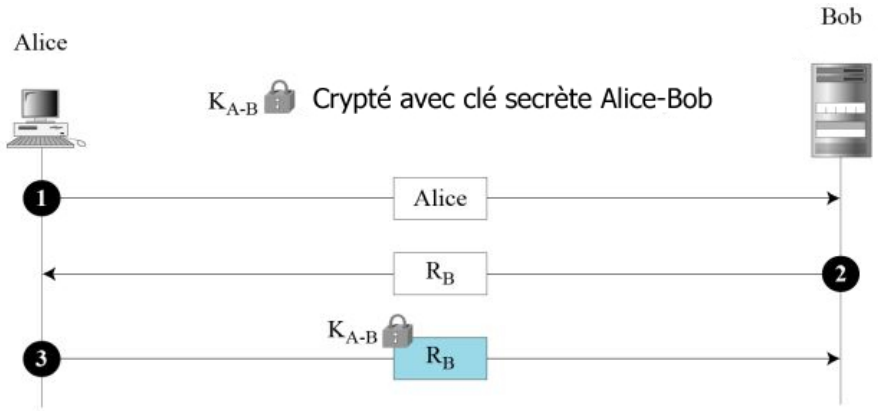


FIG. 2.3 : Authentification avec nonce

Utilisation d'une fonction de hachage à clé

Ici est utilisé essentiellement une fonction de hachage à clé (MAC). Il a l'avantage de préserver l'identité. Le défi du vérificateur est traité avec la clé secrète partagée et le client reçoit le hachage résultant. Si la réponse hachée du client correspond au résultat précédent, cela confirme une réponse correcte et l'intégrité des données. Cette couche de sécurité supplémentaire empêche l'altération des données lors de l'authentification.

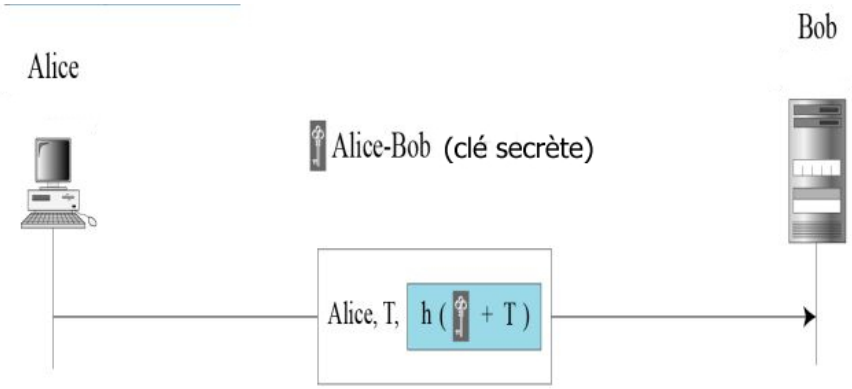


FIG. 2.4 : Authentification avec hachage

Utilisation du chiffrement à clé asymétrique

Ici, la clé privée utilisée par le client est une partie du secret. Le client doit montrer qu'il possède la clé privée, qui est liée à la clé publique clé à laquelle tout le monde a accès. Le vérificateur crypte le défi en utilisant la clé privée du client. Ensuite, le client peut déchiffrer le message en utilisant sa clé secrète et envoie un nonce. Sinon, les deux clés publiques sont utilisées chacune dans un sens.

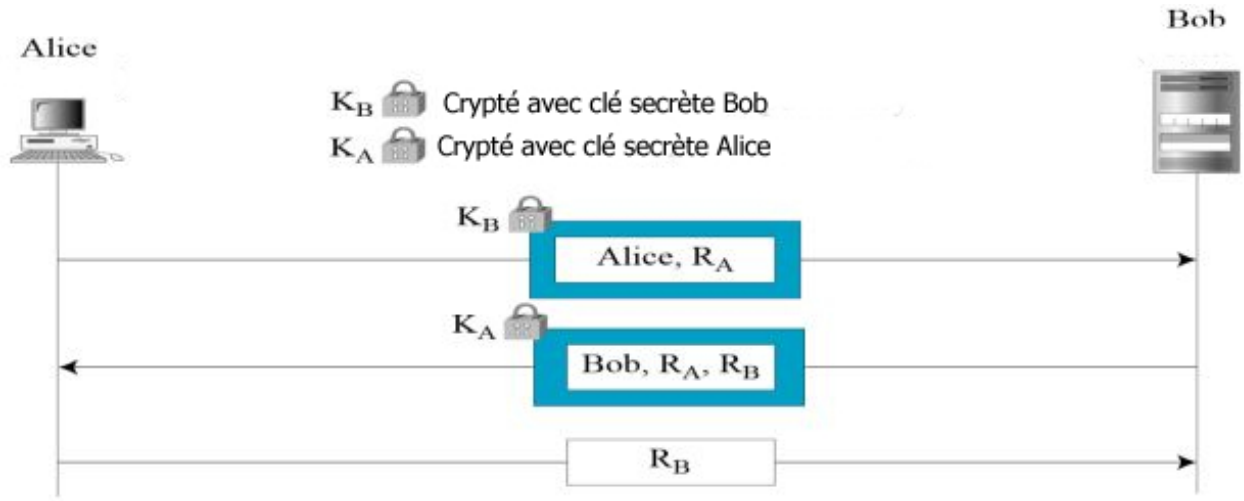


FIG. 2.5 : Défi avec MAC

Utilisation de la signature numérique

L'authentification de l'entité peut également être réalisée à l'aide d'une signature numérique. Lorsque le vérificateur reçoit les données signées, il peut vérifier l'authenticité en utilisant la clé publique du client. Cette vérification permet de s'assurer que les données proviennent bien de l'entité authentique et qu'elles n'ont pas été modifiées depuis leur signature.

La signature numérique offre une garantie solide de l'identité de l'entité et de l'intégrité des données échangées. Elle est largement utilisée dans les systèmes décentralisés, y compris notre recherche.

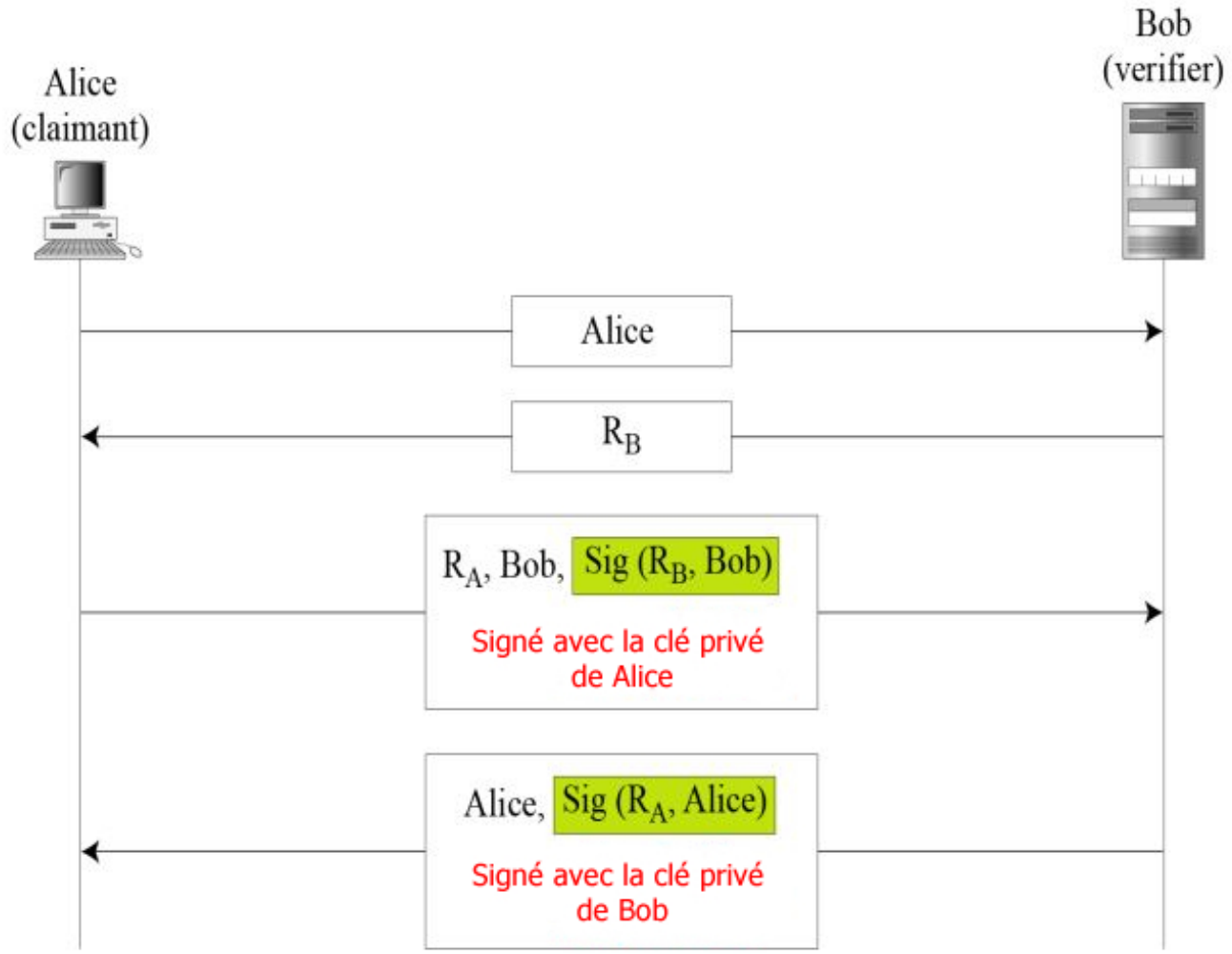


FIG. 2.6 : Défi avec signature

2.4 L'importance de l'authentification

L'authentification est essentielle pour protéger les données sensibles contre les accès non autorisés. Sans authentification, n'importe qui pourrait accéder à des informations confidentielles ou effectuer des transactions financières en votre nom.

Malgré son importance, l'authentification peut présenter des défis . L'une des principales difficultés est de trouver un équilibre entre la sécurité et la convivialité. Des mesures de sécurité trop strictes peuvent rendre l'utilisation du système difficile et frustrante pour les utilisateurs. Un autre défi est de maintenir la sécurité de l'authentification elle-même. Les pirates informatiques peuvent utiliser des techniques sophistiquées telles que le phishing pour obtenir des informations d'identification et accéder à des systèmes protégés.

Il existe une diférence entre les termes identification et authentification, mais nous les utiliserons de manière interchangeable dans notre rapport.

La principale différence entre l'identification et l'authentification est que l'authentification est le processus de reconnaissance d'un utilisateur par un système, tandis que l'identification est le processus de vérification de l'identité de l'utilisateur.

2.4.1 Relation entre authentification et preuve à divulgation nulle de connaissance

La relation entre l'authentification et la preuve à zéro connaissance est étroite, car elles peuvent être utilisées ensemble pour renforcer la sécurité des systèmes informatiques. En utilisant la preuve à zéro connaissance, il est possible de prouver son identité sans révéler d'informations sensibles. Par exemple, lorsqu'un utilisateur se connecte à un système informatique, il peut utiliser la preuve à zéro connaissance pour prouver son identité sans avoir à saisir un nom d'utilisateur et un mot de passe. Cela rend le système plus sécurisé, car les noms d'utilisateur et les mots de passe ne peuvent plus être aussi facilement piratés.

2.5 Conclusion

L'authentification est un élément crucial de la sécurité informatique . Il existe plusieurs méthodes d'authentification, chacune avec ses avantages et ses inconvénients. Il est important de choisir la méthode d'authentification appropriée en fonction des besoins de sécurité. En fin de compte, la sécurité doit être équilibrée avec la commodité pour assurer une adoption efficace de l'authentification ; Une technique efficace d'authentification en cryptographie se nomme la Zero Knowledge Proof, dont on se concentrera en détail dans le prochain chapitre.

Chapitre 3

Zero Knowledge Proof

Sommaire

3.1 Introduction

Les preuves à divulgation nulle de connaissance sont introduites pour la première fois dans les années 80, par Silvio Micali , Shafi Goldwasser et Charles Rackoff dans un papier intitulé « The Knowledge complexity of interactive proof-systems » [2].
Ils ont réussi à mettre en place un procédé qui permet de prouver une information sans révéler son contenu.
Dans ce chapitre, nous allons voir les notions de bases en voyant un exemple basique qui est la caverne d'Ali Baba. De cette base théorique , nous verrons la méthode probabiliste utilisé dans le ZKP, et dans le dernier chapitre , comment améliorer ce protocole afin qu'il soit utilisable dans un système de serveur et de client.

3.2 Notions de base sur le Zero Knowledge Proof

3.2.1 Définition du ZKP

Une preuve de connaissance nulle ou Zero Knowledge Proof (ZKP) est une méthode cryptographique permettant a un parti appelée le prouveur de prouver à un autre parti appelée le vérificateur qu'il connaît une information particulière sans révéler cette information au vérificateur. Cela peut être particulièrement utile dans des situations où la confidentialité est importante comme dans les transactions financières, l'authentification, la signature électronique et la vérification d'identité ou même la protection de la vie privée en ligne. Le ZKP marque aussi une étape importante de l'évolution de la blockchain et du Web3 notamment[4].

Le ZKP, une théorie de la complexité appliquée

Les implications pour la vie privée sont évidentes, mais le ZKP est également essentiel pour l'évolutivité. Quand on utilise une preuve à divulgation nulle de connaissance pour effectuer une tâche de calcul coûteuse, on peut généralement redémontrer la preuve sans avoir à faire un nouveau calcul. En un sens, les preuves à divulgation nulle de connaissance découlent naturellement de la théorie de la complexité et de la cryptographie. Une grande partie du chiffrement moderne (du type asymétrique) dépend de la théorie de la complexité car la sécurité asymétrique repose sur l'utilisation de fonctions a sens unique[1].

Une grande partie de la cryptographie est construite sur l'exploration des franges des mathématiques (en particulier la factorisation et le modulo) pour trouver des propriétés utiles. On peut faire des milliers de choses intéressantes avec ce genre de fonctions à sens unique[5]. En particulier, on peut établir des secrets partagés sur des réseaux ouverts, une capacité sur laquelle reposent les communications sécurisées modernes.

Le Zero Knowledge Proof pose la question suivante : **est-il possible d'utiliser ces types de calculs pour prouver quelque chose tout en gardant l'information cachée ?**

3.2.2 La caverne d'Ali Baba

Un exemple connu du Zero Knowledge Proof est celui de la caverne d'Ali Baba[6]. Une caverne se divise en deux chemins, A et B. Reliant ces deux chemins, une porte qui ne s'ouvre qu'avec un code secret ; "Ouvre toi sésame" . Dans l'exemple de la caverne d'Ali Baba, ce code secret permet de passer d'un chemin a l'autre, mais impossible d'ouvrir la porte sans connaître le code secret.

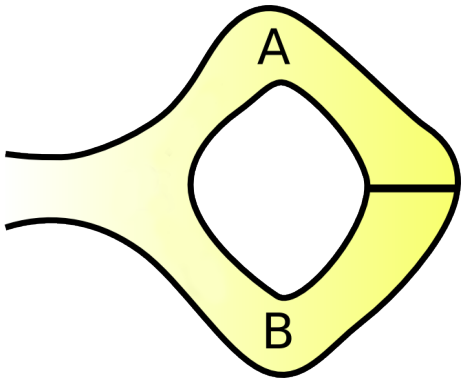


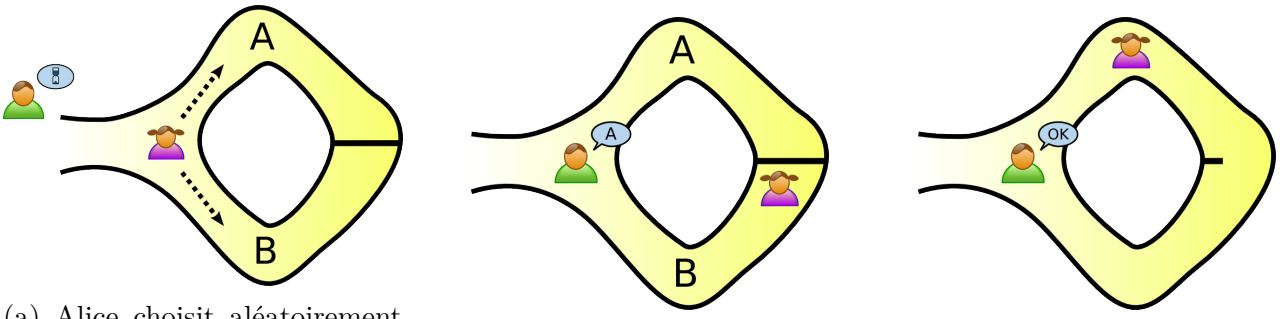
Fig. 3.1 : La caverne d'Ali Baba

Alice connaît le code secret, elle veut prouver a Bob qu'elle connaît le code secret, sans révéler le code. En d'autre terme, elle veut le convaincre qu'elle connaît un code secret la permettant d'aller de A vers B. Pour ce faire Alice et Bob vont répéter plusieurs fois le scénario suivant (voir fig 3.2) :

- Bob attend à l'entrée de la caverne et ne voit pas l'intérieur du tunnel. Alice s'introduit dans la galerie et choisit aléatoirement le chemin à gauche A ou à droite B.
- Bob entre dans le tunnel et attend au croisement .Il choisit un chemin et crie A ou B.
- Alice doit prouver maintenant qu'elle est en possession du code secret ,elle doit apparaître vers la sortie demandée par Bob.

Si Alice connaît le mot magique, elle apparaîtra toujours là ou Bob lui demande d'apparaître, sans jamais révéler le secret. Bob aura donc la preuve qu'il existe bien un code permettant de passer de A vers B.

Une erreur d'Alice sera considérée comme une preuve de non-connaissance. Bob peut de suite arrêter, il est assuré qu'Alice ne connaît pas le mot magique. Cependant, il y a un autre cas de figure. Il se peut que Alice mente, et qu'elle ait été chanceuse. Elle a 50 % de chance de bien se



(a) Alice choisit aléatoirement le chemin A ou B ; Bob attend dehors

(b) Bob crie un des deux chemins (A ou B)

(c) Alice apparaît sur le chemin demandé par Bob

FIG. 3.2 : Déroulement du scénario

situer là ou Bob voulait, sans avoir le mot de passe. C'est pour cela que l'opération est répétée plusieurs fois. À chaque nouvel essai, il améliore son assurance.

Si Alice n'est pas en possession du mot magique, il y a $\frac{1}{2}$ de chance pour qu'elle commette une erreur. Avec N essais, la probabilité pour que Bob affirme qu'Alice possède le secret alors qu'elle ne le possède pas est de $\frac{1}{2^n}$, n étant le nombre de répétitions. Plus il sera élevée, plus la probabilité sera faible. En répétant le test 20 fois par exemple , la probabilité sera de $\frac{1}{1\,048\,576}$

3.2.3 Propriétes

Trois propriétés clés doivent êtres satisfaites afin de constituer une preuve a divulgation nulle de connaissance

1. Consistance : Un prouveur honnête, qui suit le protocle correctement, passera toujours le test.

2. Robustesse : un prouveur malveillant échouera au test avec une forte probabilité.

3. Aucun apport d'information : le vérificateur n'apprend de la part du fournisseur de preuve rien de plus que la véracité de l'information, il n'obtient aucune information additionnelle.

Dans l'exemple précèdent, le protocole qu'utilise Alice vérifie ces trois propriétés. Si elle connaît le mot de passe, elle passera toujours le test. Si elle ne connaît pas le mot de passe, elle échouera le test au bout du compte. Et finalement Bob n'apprendra jamais le mot de passe secret, seule la véracité du mot de passe est démontrée.

3.2.4 Utilisations

Les utilisations du Zero Knwoledge Proof sont nombreuses. En effet, tout problème dans la classe NP a une preuve a divulgation nulle. Pour rappel, La classe NP est une classe très importante

de la théorie de la complexité. L'abréviation NP signifie non déterministe polynomial. C'est a dire qu'on peut vérifier rapidement si une solution candidate est bien solution. Il y a des tonnes d'applications futures ou vérifier une solution secrète sans la révéler serait utile.

- Garantir la sécurité des mots de passes, éliminant ainsi toute tentative de Phishing
- Dans les cryptomonnaies, ou chaque transaction est publique et visible
- La divulgation sélective ou partielle de données
- Apport de preuve qu'une certaine portée est bien respectée
- Preuve de droit d'auteur sur du contenu web
- etc ...

3.3 Démonstration probabiliste

Probleme du logarithme discret

Le logarithme discret est un objet mathématique utilisé en cryptographie ,il est utilisé pour la cryptographie à clé publique. Soit g et y des nombres entiers, et p un large nombre premier. La connaissance de ces valeurs sont publiques . Nous voulons prouver d'existence d'un nombre entier x , sans révéler sa valeur tel que :

$$y = g^x mod\ p$$

Prenons un exemple ou g=3, p=5, y=1. Il s'agit de trouver x qui satisfait cette formule tel que

$$3^x mod\ 5 = 1$$

L'algorithme du logarithme discret n'est solvable que par l'essai et l'erreur, jusqu'à trouver un x qui satisfait l'énoncé. Dans notre exemple ;

$$3^1 mod\ 5 = 3$$

$$3^2 mod\ 5 = 4$$

$$3^3 mod\ 5 = 2$$

$$3^4 mod\ 5 = 1$$

Donc x=4, après avoir essayer toutes les valeurs possibles, une par une.

Mais pour de grands nombres g et p , cela pourrait prendre avec des machines performantes

des années de calcul avant de trouver le x qui satisfait l'énoncé [7]. Cela pourrait prendre théoriquement des centaines de milliers d'années pour craquer une clé x de 1024 bits par exemple.Ceux de 2048 bits prendraient 16 millions de fois plus de temps a craquer [8]. Prouver la connaissance d'un ensemble x,g et p reste irréalisable pour quiconque n'a pas la réponse en amont. Le calcul de vérification est facile mais seule la personne connaissant la valeur x pourrait démontrer sa satifiabilité. Tandis que la probabilité de trouver x sans le connaître auparavant est infiniment petite.

Utilité

Démontrer la satisfiabilté de x revient à prouver la connaissance de la valeur x, sans révéler sa vraie valeur. Cela pourrait être utile si x est un mot de passe par exemple. Donc démontrer que x est connu, seulement par l'utilisateur, sans l'envoyer directement au vérificateur. On en revient donc au Zero Knowledge Proof.

3.4 Diffie Hellman Key Exchange

L'échange de clés Diffie-Hellman est basé sur la difficulté de résoudre le problème du problème du logarithme discret, c'est-à-dire qu'il est difficile de calculer x à partir de $g^x mod\ p$

En supposant que le problème du logarithme discret est informatiquement irréalisable [7], un attaquant espionnant par hasard la conversation entre Alice et Bob ne peut pas apprendre $g^x mod\ p$

1. D'abord Alice génère r un nombre entier aléatoire, privé, mais envoie $u{=}g^r mod\ p$ une valeur publique ainsi que $v = x + r$,

2. Bob peut alors vérifier si $u * y = g^v mod\ p$

Car on sait que

$$g^x * g^y = g^{x+y}$$

Donc on a

$$g^r mod\ p * g^x mod\ p = g^{x+r} mod\ p$$

Sans avoir accès a x, il est possible de vérifier sa cohérence a partir de y et d'un nombre $v = x + r$

3.4.1 Premier problème

Un premier problème d'ordre sécuritaire se pose. Si Alice est malveillante, elle peut envoyer un $u = \frac{g^r}{y} mod\ p$ Cette formule peut satisfaire le test, alors même que la valeur de x n'est pas connue :

$$y * \frac{g^r}{y} mod\ p = g^r mod\ p$$

Puisque Alice a elle même choisie la valeur r, il lui est donc facile de prouver x sans même connaître sa valeur. Donc elle ne satisfait pas la seconde propriété du Zero Knwoldege Proof, a savoir la Robustesse.

Solution

Bob demande r. Ainsi il peut vérifier que $u{=}g^r mod\ p$, et non un u qui a été fabriqué pour détourner le test

3.4.2 Deuxième problème

Si Bob a accès a r et $r + x$, il peut en déduire la valeur de x, donc ce protocole ne satisfait pas la troisième propriété du Zero Knowldge Proof, l'apport d'information. X n'est plus un secret.

Solution

La solution est de ne pas demander r et $r + x$, mais seulement l'un des deux, aléatoirement. Comme pour la caverne , où Bob testait A ou B plusieurs fois. Même si Alice est malveillante , elle ne pourra pas se préparer au test A ou B a l'avance, elle aura $\frac{1}{2}$ chance a chaque itération, jusqu'à arriver à une probabilité infinitésimale.

Donc comme vu auparavant, pour sécuriser un Zero Knowledge Proof, il faut faire un test aléatoire plusieurs fois, jusqu'à arriver à une probabilité faible que Alice soit malveillante. C'est pour cela que le Zero Knowledge Proof n'est pas une preuve mathématique, mais une preuve probabiliste.

3.5 Protocole intéractif

Algorithme 1 : Algorithme du protocle intéractif

1 `Debut :`

2 Alice publie $g^x mod\ p\ =\ y$

3 Alice choisie aléatoirement un nombre
Alice envoie a Bob $u{=}g^r mod\ p$

4 Maintenant, Bob a un artefact basé sur ce nombre aléatoire, mais ne peut pas réellement calculer le nombre aléatoire
Bob envoie un défi e, soit 0 ou 1

5 Alice répond avec u :
Si 0, $v = r$
Si 1, $v = u + x\}$

6 Bob peut maintenant vérifier :
Si 0: Bob a le nombre aléatoire r, ainsi que les variables publiquement connues et peut vérifier si $u{=}g^v mod\ p$
Si 1: $u * y{=}g^v mod\ p$

7 **Fin ;**

8

Si r est vraiment aléatoire, réparti entre 0 et $p - 1$, cela ne laisse aucune fuite d'informations sur x, ce qui est assez ingénieux, mais pas suffisant.

Afin de s'assurer que x ne peut pas être usurpée, plusieurs itérations sont nécessaires ainsi que l'utilisation de grands nombres. Le protocole satisfait toutefois les propriétés du Zero Knowledge Proof :

- Consistance : Si Alice connaît x, elle répondra toujours correctement.
- Robustesse : Sinon $\frac{1}{2^n}$ chance de tricher le défi, n étant le nombre de répétitions.
- Aucun apport d'information : Au final, Bob ne connaît toujours pas x, mais il est sûr que x est bien en la possession d'Alice.

3.5.1 ZKP interactif

Le protocole décrit précédemment est un ZKP interactif. Il demande une interaction constante entre le prouveur et le vérificateur, pendant plusieurs tours. Les ZKP interactifs présentent quelques problèmes[9].

Tout d'abord, ils sont souvent coûteux en temps de calcul, car ils nécessitent plusieurs itérations, ce qui peut être taxant sur le serveur . De plus, les ZKP interactifs peuvent être vulnérables aux attaques de type "man in the middle", où un attaquant intercepte les échanges entre le prouveur et le vérificateur et modifie les messages pour compromettre la sécurité du protocole.

3.6 Les alternatives au ZKP interactif

Il existe plusieurs alternatives aux preuves de connaissance nulle interactives :

- Preuves de connaissance nulle non interactives (Non-Interactive Zero Knowledge Proofs, NIZK) : les NIZK permettent à une partie de prouver la connaissance d'une information sans interaction avec une autre partie. Elles sont utilisées pour prouver la possession de clés secrètes dans les protocoles cryptographiques.
- Preuves de connaissance argumentées (Argumented Zero Knowledge Proofs, AZKP) : les AZKP permettent à une partie de prouver la connaissance d'une information en fournissant une preuve qui peut être vérifiée par une autre partie. Elles sont utilisées pour prouver l'authenticité d'un message ou d'une identité.
- Arguments succincts non interactifs à connaissance nulle (SNARG) : atteindre la concision au prix d'hypothèses cryptographiques plus solides et de garanties de sécurité souvent plus faibles.
- Arguments de connaissance succincts non interactifs à connaissance nulle (SNARK ou parfois zk-SNARK) : ce sont des SNARG qui sont aussi des preuves de connaissance et de connaissance nulle.
- Arguments de connaissance transparents succincts (STARK) : transparent fait ici référence à la configuration ne nécessitant qu'une fonction de hachage de confiance. Ceci est bénéfique mais peut entraîner une surcharge de performances.

Chacune de ces alternatives présente des avantages et des inconvénients en termes de sécurité, d'efficacité et de facilité d'utilisation.

Actuellement, le système de preuve le plus attrayant du point de vue du vérificateur est " argument de connaissance succinct non interactif " ou **zk-SNARK** en abrégé, qui a une petite

taille de preuve constante et des coûts de vérification à temps constant même pour des quantités arbitrairement grandes.

3.7 Qu'est-ce qu'un zk-SNARK ?

L'acronyme zk-SNARK signifie "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge" Le protocole zk-SNARK utilise des mathématiques avancées telles que la théorie des nombres algébriques et la géométrie algébrique pour construire des preuves cryptographiques succinctes et vérifiables, sans avoir besoin de révéler toutes les informations associées à la transaction.

Plus précisément, le protocole est une fonction de hachage pour créer une empreinte numérique. Cette empreinte est ensuite utilisée pour générer un zk-SNARK, qui par la suite est envoyé pour validation.

Le processus de génération d'un zk-SNARK implique la création d'un circuit arithmétique qui représente l'argument en question. Ce circuit arithmétique est ensuite converti en un système de contraintes qui peut être utilisé pour générer un ensemble de polynômes. Ces polynômes sont ensuite utilisés pour générer la preuve succincte qui est envoyée au vérifieur pour validation. C'est le système de contraintes R1CS

3.7.1 Système de contraintes R1CS

De nombreux systèmes de preuve différents liés à zkSNARK ont été inventés, et il s'avère que la plupart d'entre eux s'appuient sur un protocole appelé "Rank-1 contraint satisfiability" (R1CS) qui peut être introduite dans le système de preuve Ainsi, le protocole R1CS est un moyen puissant de capturer formellement un circuit algébrique et de le traduire en un ensemble de matrices et de vecteurs pouvant être introduits dans un système de preuve en aval.

Pour notre exemple, l'équation à démontrer est $x^3+x+5=35$, qui est satisfaite via x=3 en un ensemble de portes logique, puis en le faisant passer par le protocole R1CS.

On commence par l'aplatissement, en transformant l'équation $x^3+x+5=35$ en une séquence d'équations plus simples de portes arithmétiques qui ont chacune au plus une opération de multiplication :

$$sym1 = x * x$$

$$y = sym1 * x$$

$$sym2 = y + x$$

$$sortie = sym2 + 5$$

Ensuite, nous convertissons la tâche de calcul en quatre équations :

$$9 = 3 * 3$$

$$27 = 9 * 3$$

$$30 = 27 + 3$$

$$35 = 30 + 5$$

Il est important de noter que si nous pouvons prouver l'exactitude de ces 4 équations, cela équivaut à prouver l'équation d'origine, $x^3+x+5=35$ via $x=3$

Nous pouvons maintenant mettre en place les portes logiques du système de contraintes R1CS, qui sont comme suit pour l'équation :

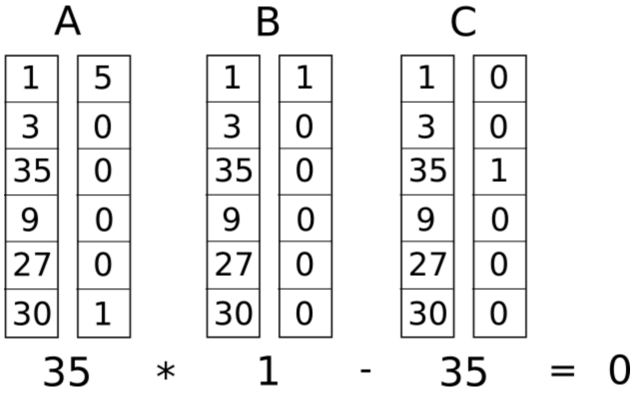


FIG. 3.3 : Les portes logiques dy système d'équation

Les valeurs des vecteurs triplés pour la première porte , donc la solution que nous recherchons sont :

$$a = [0, 1, 0, 0, 0, 0]$$

$$b = [0, 1, 0, 0, 0, 0]$$

$$c = [0, 0, 0, 1, 0, 0]$$

3.7.2 Différence entre Preuve et Argument

Précédemment nous utilisons le mot preuve pour décrire un ZKP, mais dans les zk-SNARKS, nous utilisons le terme argument. Il existe une différence entre les deux termes. Un argument de connaissance nulle est tout simplement une preuve de connaissance nulle qui a une solidité informatique plutôt qu'une solidité statistique[10].

En général, les preuves ne peuvent avoir qu'une connaissance nulle en calcul, tandis que les arguments peuvent avoir une connaissance nulle parfaite.

3.7.3 Les avantages

L'utilisation de zk-SNARKs offre plusieurs avantages par rapport aux méthodes de preuve traditionnelles telles que les preuves interactives à divulgation de connaissance (IZKP)

- Efficacité : Les zk-SNARKs sont beaucoup plus efficaces que les IZKPs en termes de temps de calcul et de taille de preuve. Les zk-SNARKs peuvent être vérifiés en quelques millisecondes, tandis que les IZKPs nécessitent une communication en temps réel entre le prouveur et le vérificateur, ce qui peut prendre plusieurs secondes ou minutes.

- Confidentialité : Les zk-SNARKs offrent un niveau supérieur de confidentialité car ils ne nécessitent pas de communication en temps réel entre le prouveur et le vérificateur. Le prouveur peut simplement envoyer la preuve zk-SNARK au vérificateur, qui peut alors vérifier la preuve sans connaître l'information elle-même.

- Taille de preuve : Les zk-SNARKs produisent des preuves beaucoup plus petites que les IZKPs. Cela peut être important dans des situations où l'espace de stockage est limité ou lorsque les preuves doivent être transmises sur des réseaux à faible bande passante.

- Utilisation pratique : Les zk-SNARKs ont été utilisés dans des applications pratiques, telles que la cryptomonnaie ZCash [11] pour fournir des transactions confidentielles. Les IZKPs sont généralement utilisés dans des applications où le temps de calcul n'est pas un facteur critique.

3.7.4 Les signatures de Shnorr

La signature de Schnorr est un algorithme de signature numérique inventé par le cryptographe allemand Claus-Peter Schnorr en 1989. Cette méthode de signature est considérée comme l'une des plus efficaces et des plus sûres parmi les algorithmes de divulgation nulle de connaissances non interactifs[12].

L'idée de base de la signature de Schnorr est d'utiliser une clé privée pour générer une signature pour un message, et une clé publique pour vérifier cette signature. Contrairement à d'autres algorithmes de signature numérique, la signature de Schnorr utilise une fonction de hachage pour réduire la taille du message à signer, ce qui la rend plus rapide et plus efficace.

Dans la configuration du système, Alice publie sa clé publique $A = G^a$, où a est la clé privée choisie uniformément au hasard de [1, n-1].

Le protocole fonctionne en trois passes :

Algorithme 2 : Algorithme du protocole de Schnorr

1 `Début :`

2 Alice choisit un nombre v uniformément au hasard parmi [1, n-1] et calcule $V = G^v$. Elle envoie V à Bob.

3 Bob choisit un défi c uniformément au hasard parmi [0, 2(t-1)], où t est la longueur en bits du défi (par exemple, t = 80). Bob envoie c à Alice.

4 Alice calcule r = v - a * c mod n et l'envoie à Bob.)

5 À la fin du protocole, Bob effectue les vérifications suivantes. Si toute vérification échoue, le processus est interrompu et elle n'est pas authentifié.

6 Pour vérifier que A est un point valide sur la courbe et que A^h ne pointe pas à l'infini ; Vérifier$V = G^r + A^c$

7 **Fin ;**

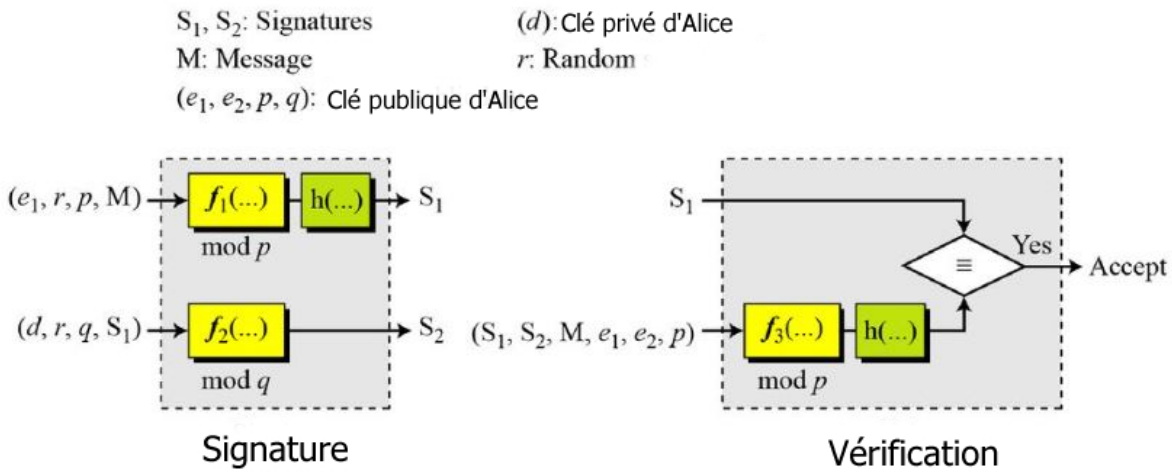


FIG. 3.4 : Le Protocole de Schnorr

3.7.5 Heuristique de Fiat-Shamir

Le protocole d'identification dérivé par l'heuristique de Fiat-Shamir est une technique couramment utilisée pour authentifier des utilisateurs dans des systèmes cryptographiques. Ce protocole est dérivé du protocole d'identification interactif de Schnorr.

Dans le protocole d'identification dérivé par l'heuristique de Fiat-Shamir, la deuxième phase du protocole d'identification interactif de Schnorr est remplacée par une fonction de hachage unidirectionnelle. Un tiers parti choisi au hasard des nombres p et q pour vérifier que n=p*q. p et q sont tenues secrètes. Alice choisit un nombre mystère compris entre 1 et n-1. Elle constate $v = s^2 \mod p$. Elle garde s comme sa clé privée et enregistre v comme sa clé publique .

Algorithme 3 : Algorithme de l'heuristique de Fiat Shamir

1 `Debut :`

2 Alice, choisit un nombre entre 0 et n-1, et r étant appelé l'engagement. Elle peut évaluer la valeur de $x = r^2 mod\ n$; et nommer la valeur x comme témoin.

3 La valeur de x est envoyée à Bob par Alice comme témoin.

4 Ensuite, Bob envoie une valeur de c, qui est appelée l'engagement. Ça peut être soit 0 soit 1.

5 Alice évalue la réponse $y = rs^c$

6 Alice envoie la valeur de la réponse y et prétend être Alice.

7 Bob calcule y^2 et xv^c

8 Si ces deux valeurs conviennent, alors Bob peut conclure qu'Alice connaît la valeur de la clé secrète ou qu'elle a deviné la valeur de y d'une quelconque manière malhonnête, ce qui peut probable.
Fin ;

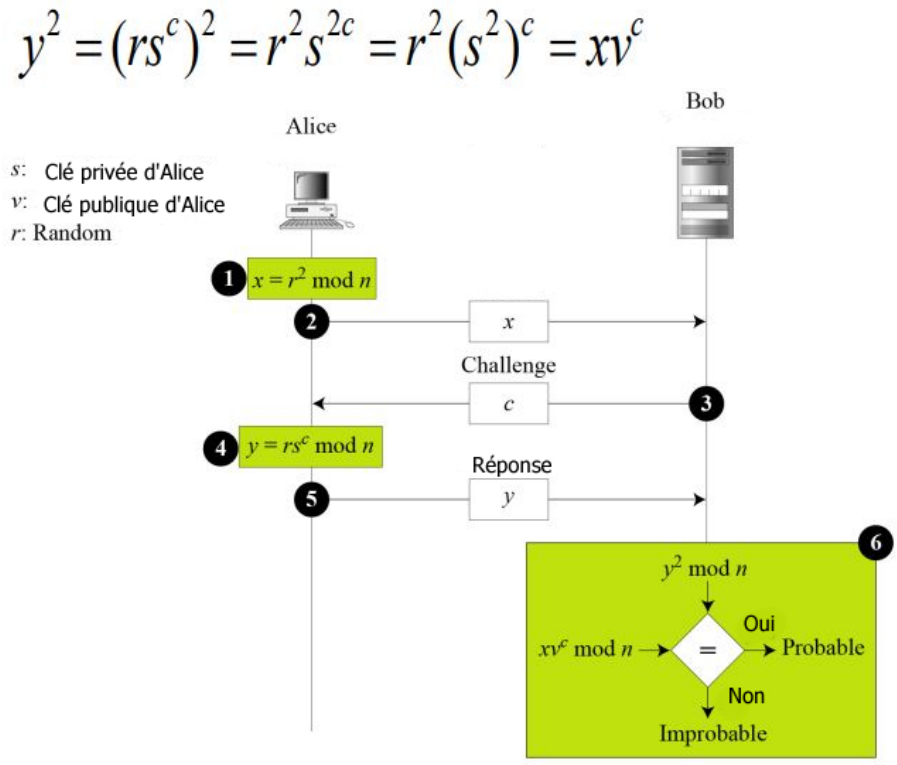


FIG. 3.5 : Le Protocole de Fiat-Shamir

Le protocole d'identification dérivé par l'heuristique de Fiat-Shamir est plus efficace que le protocole d'identification interactif de Schnorr [13], car il ne nécessite qu'une seule communication entre le prouveur et le vérificateur, contrairement à deux pour le protocole d'identification interactif de Schnorr.

De plus, il est résistant aux attaques par rejeu (replay attack) , car les valeurs aléatoires sont utilisées pour chaque exécution du protocole. Cependant, il est important de noter que le protocole d'identification dérivé par l'heuristique de Fiat-Shamir est vulnérable à certaines attaques, telles que les attaques par collision de hachage.

3.7.6 Protocole Guillou-Quisquater

Le protocole Guillou-Quisquater est une extension de la convention Fiat-Shamir dans lequel moins de tours sont utilisés pour identifier l'utilisateur. Dans ce protocole, une troisième partie de confiance (parfois appelée autorité de certification ou "CA") est utilisée pour garantir la sécurité

et l'authenticité du protocole.

Algorithme 4 : Algorithme de l'heuristique de Guillou Quisquater

1 `Début :`

2 La troisième partie est utilisée pour générer les informations secrètes pour les deux parties, elle choisit deux nombres premiers p et 1 pour calculer le n=p*q.

3 Le CA déclare v qui sert de clé publique $v = pgcd(v, \theta(n))) = 1, \theta(n)$ étant le copremier de n

4 Le CA déclare un nombre s qui sert de clé privée tel que sv=1 mod $\theta(n)$

5 Alice choisit un nombre aléatoire r et calcule $x = r$ mod n

6 Alice envoie x et J_A *a Bob*

7 Bob choisit un nombre aléatoire e et l'envoie a Alice

8 Alice calcule $y = rS_A^e \ mod \ n$ et l'envoie à Bob

9 Bob calcule $J_A^e y^v$ et vérifie que le résultat est égal à x et différent de 0.

10 **Fin ;**

11

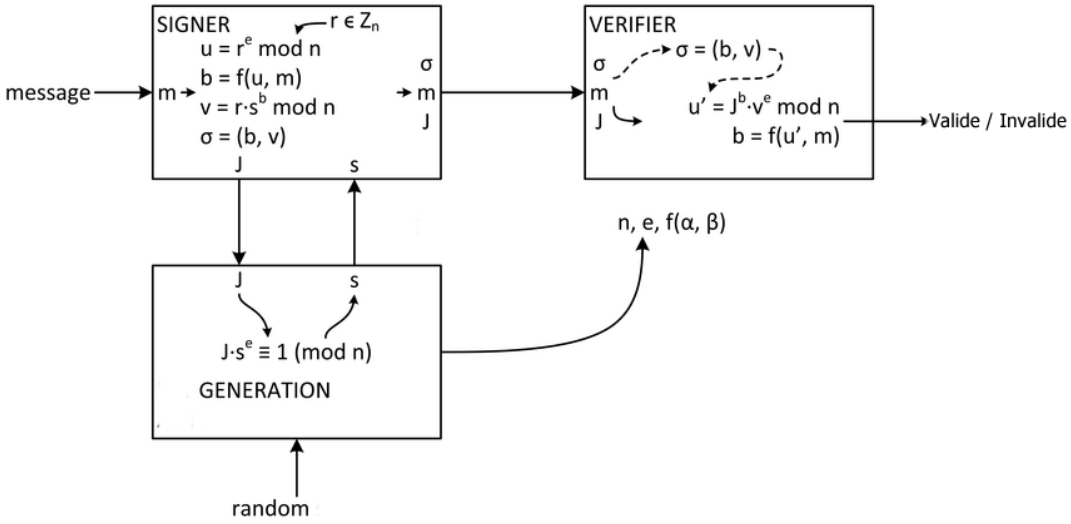


FIG. 3.6 : Le Protocle de Guillou-Quisquater

3.8 Les système de gestions d'identité

Dans l'écosystème numérique aujourd'hui, toute personne et tout objet est associé à une identité numérique. Les identités numériques permettent aux systèmes, aux services et aux applications de savoir avec qui ils interagissent.

Alors que le nombre de personnes, d'appareils, de services et d'objets continue de croître de manière exponentielle, il est essentiel de gérer l'intégralité du cycle de vie des identités et de s'assurer qu'une autorisation d'accès adaptée est accordée à la bonne personne, au bon appareil ou au bon objet.Cela nécessite une plateforme de gestion des identités complète et dynamique, capable de proposer un niveau de sécurité, de confidentialité et de conformité sans compromis, et d'offrir une expérience fluide et facilitée aux utilisateurs.

Les DIMS, ou systèmes de gestions d'identité, prennent en charge la gestion des identités et des accès pour tous les cas d'utilisation. Mais il existe quelques défauts flagrants a ces systèmes, notamment la vulnérabilités de sécurité , vulnérables aux attaques et aux violations de sécurité.Ils peuvent être exploités par des individus malveillants pour accéder à des informations sensibles ou voler des identités.

Un deuxième problème est la confidentialité : Les systèmes de gestion des identités stockent souvent une quantité considérable d'informations personnelles, telles que des noms, des adresses, des numéros de sécurité sociale, ce qui rend le tout non anonyme.

3.9 SSI

Après avoir acquis des notions sur les DIMS et leurs incompatibilité avec l'écosystème numérique courant, voyons une alternative qui entrave moins la vie privée des utilisateurs ; les SSI , Self-Sovereign Identity ou identité Auto-souveraine. C'est une approche où l'individu contrôle et gère l'accès à son identité numérique, sans l'intervention d'une partie tierce

Cette approche se veut basé sur l'utilisateur, puisqu'elle suppose que l'individu a les pleins pouvoirs sur la gestion de ses données personnelles.

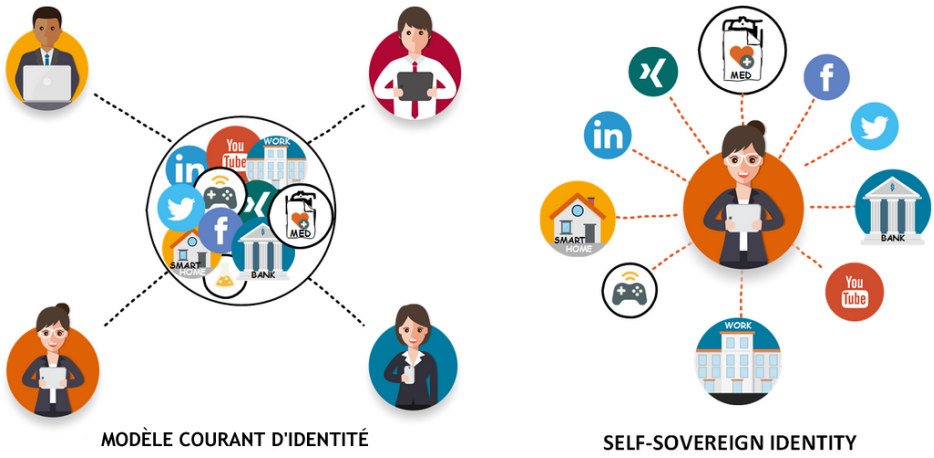


FIG. 3.7 : Schéma comparatif des modèles d'identité

Un tel système nécessiterait néanmoins d'être totalement sur et confidentiel [14]. Quelques recherches ont été faites sur l'implémentation d'un système ZKP sur cette base, et ce que nous verrons dans le prochain chapitre.

3.10 Conclusion

Après avoir acquis les bases sur lesquelles repose un cryptosystème conçu par le ZKP, nous pourrions comprendre les avantages qu'offre un tel système comme la sécurité malgré la simplicité de ses opérations de base, ce qui est un point très fort vis à vis des environnements modernes à sécuriser tel que : les réseaux, les bases de données, le cloud, les objets connectés qui présentent des contraintes variées, tel que la bande passante et l'énergie, ce qui exige moins de calculs et des clés moins volumineuses sans perte dans le niveau de sécurité. Dans le chapitre suivant, nous allons aborder un état de l'art, où nous décrivons quelques travaux dans le domaine du ZKP, puis ferons une synthèse pour avoir une vue d'ensemble.

Chapitre 4

État de l'art

Sommaire

4.1 Introduction

Une variété de travaux de recherche ont été effectué sur le Zero Knowledge Proof depuis la parution de l'article de Goldwasser dans les années 80.
Depuis lors, plusieurs protocoles d'authentification ont vu le jour, notamment ceux basés sur les zk-SNARKS, visant à minimiser le temps d'exécution des algorithmes tout en réduisant la taille des preuves et sans négliger le niveau de sécurité des systèmes établis.
Dans cet état de l'art, nous présentons plusieurs approches apportant chacune une utilisation unique et viable, qu'on a proposé d'exposer suivant les principaux protocoles examinés.

4.2 Identité digitale basé sur ZKP

Les systèmes traditionnels de gestion centralisée de l'identité numérique ont été sujet de plusieurs menaces . La technologie émergente de la blockchain a permis d'atténuer les problèmes posés par les tiers centralisés, mais sa transparence inhérente et le manque de confidentialité posent de grands défis. Pour résoudre ces deux problèmes, une variété de recherches sur la création et la gestion de l'identité auto-souveraine (SSI) en implémentant le Zero-Knowledge Proof (ZKP) ont été mené permettant de gérer plusieurs identités, leurs identifiants et leurs attributs pour générer des preuves.

4.2.1 MOHAMEDEN DIEYE et al. 2016

Le protocole SSI proposé dans cette recherche [15] ne réutilise pas la même clé secrète que dans le cas du protocole ZKP Schnorr . Le protocole conçu garantit une divulgation minimale d'informations à un seul tiers de confiance. De plus, il ne permet aucune divulgation d'informations aux fournisseurs de services nécessitant une preuve d'identification.

- Phase d'initialisation : Alice choisit un x qu'elle veut garder secret appartenant au groupe Z_q^* puis calcule y = g^x en utilisant un générateur g d'un groupe G d'ordre q, où q est un nombre premier. Ici, on suppose que g et q ont été défini comme des variables publiques, donc Toutes les personnes impliquées dans l'accord le savent.

  Alice envoie (x, y) paires à Sam

  Sam vérifie si le x reçu n'a pas déjà été reçu

  Sam renvoie alors un couple (y, HS) à Alice comme confirmation, où HS est la fonction de hachage utilisée pour calculer y' = $g^{HS(x)}$

- Phase d'enregistrement : En utilisant le ZKP de Schnorr, Sam génère H et r qui est l'information d'identité générée par Alice, puis t la clé publique avec état, et T qui combine les informations de la clé publique d'Alice enregistrés (H, r, g, q, T) dans la blockchain.

- Phase de vérification : Alice prouve à Bob qu'elle connaît x en fournissant à B avec un couple (y ;y') lié à son existence

  L'identité d'Alice ainsi que le décor (H', R', g, q, t') lié à la preuve de connaissance par Alice x en utilisant ZKP Schnorr.

  Bob, effectuant l'étape de vérification du protocole de Schnorr, entre dans la blockchain et vérifie Que T tel que T*$y^{-}1$ = t*t'

  Bob a ensuite vérifié que la paire unique d'Alice (x, y) est bien une paire approuvée et attribuée par Sam à Alice sans savoir x

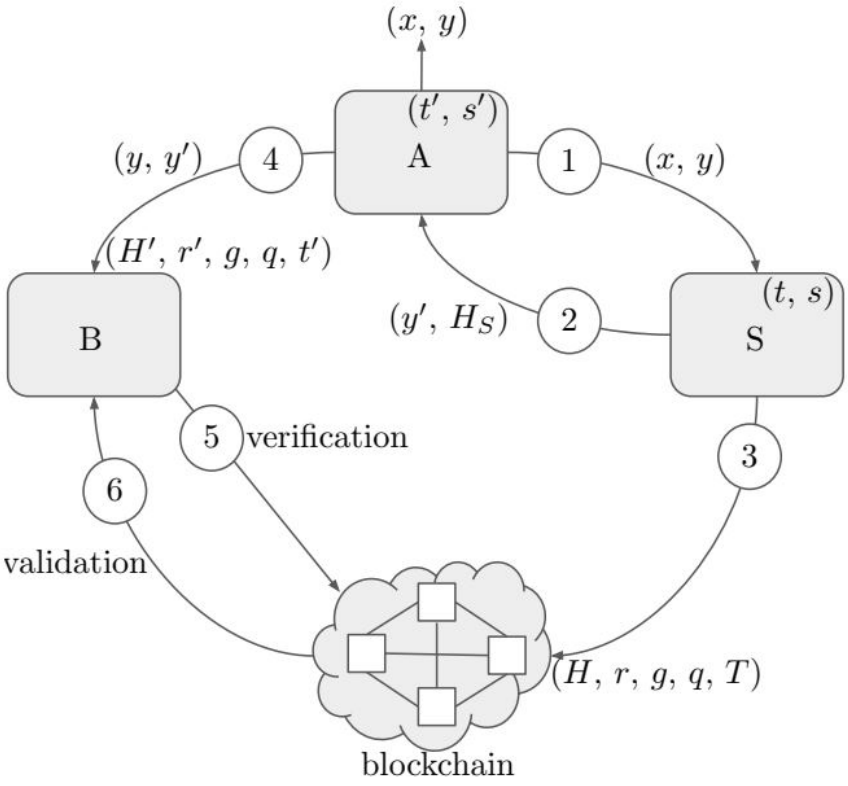


FIG. 4.1 : Schéma de vérification

4.2.2 Xiaohui Yang et al. 2020

Dans cette recherche [16], une implémentation d'un prototype de système nommé BZDIMS comprend un protocole de défi-réponse, qui permet aux utilisateurs de divulguer sélectivement leur attributs aux fournisseurs de services pour protéger la confidentialité du comportement des utilisateurs. Ils démontrent aussi un champ d'application plus large par rapport au modèle précédent.

- Phase de Création : la création est la première étape dans laquelle un jeton qui contient une

paire nom-valeur est publié puis enregistré sur l'adresse du créateur en tant qu'approbation de l'attribut.

- Phase de Transfert : Pour transférer secrètement la possession de la propriété attribut vacant à l'utilisateur dans la blockchain, le fournisseur d'identité doit émettre le jeton hashé et prouver qu'il l'a effectué par une preuve nommé claimPK qui est haché à partir de l'identifiant du jeton et de la clé publique ZK de l'utilisateur par ZKP.
- Phase de Réponse : Pour démontrer la possession d'un jeton claimPK, l'utilisateur doit prouver qu'il possède la clé privée ZK correspondant à la clé publique ZK contenue dans la préimage de hashclaim. Le calcul particulier par l'utilisateur au cours de ce processus est nommé responseSK. Cette étape fait également partie du protocole de défi-réponse.

FIG. 4.2 : Schéma du protocole BZDIMS

4.3 Hachage spécifique pour les ZKP

De nombreux cas d'utilisation des Zero Knowledge Proof implique la connaissance d'une préimage sous une certaine fonction de hachage cryptographique, qui s'exprime sous la forme d'un circuit sur un grand champ de nombre premier, mais l'insuffisance du SHA-256 comme fonction de hachage pour un tel circuit cause une grande pénalité de calcul.

4.3.1 Lorenzo Grassi et al. 2020

Dans le cadre de leurs recherches [17], Lorenzo Grassi et al. Créent une fonction qui fonctionnent nativement avec les R1CS , cette fonction nommé POSEIDON utilise jusqu'à 8 fois moins de

contraintes par bit de message que les hachages précédents dans les contextes de ZKP. apportant ainsi un nouvel élan à la performance.

POSEIDON-128				
Arity	Width	R_F	R_P	Total constraints
2:1	3	8	57	7290
4:1	5	8	60	4500
8:1	9	8	63	4050
Rescue-x^5				
2:1	3	16	-	8640
4:1	5	10	-	4500
8:1	9	10	-	5400
Pedersen hash				
510	171	-	-	41400
SHA-256				
510	171	-	-	826020
Blake2s				
510	171	-	-	630180
MiMC-2p/p (Feistel)				
1:1	2	324	-	19440

(a) Benchmark de POSEIDON sur le nombre de contraintes

Field	Arity	Merkle 2^{30}-tree ZK proof Bulletproofs time		R1CS constraints
		Prove	Verify	
POSEIDON-128				
BLS12-381	2:1	16.8s	1.5s	7290
	4:1	13.8s	1.65s	4500
	8:1	11s	1.4s	4050
BN254	2:1	11.2s	1.1s	7290
	4:1	9.6s	1.15s	4500
	8:1	7.4s	1s	4050
Ristretto	2:1	8.4s	0.78s	7290
	4:1	6.45s	0.72s	4500
	8:1	5.25s	0.76s	4050
SHA-256 [BBB+18]				
$GF(2^{256})$	2:1	582s	21s	762000

(b) Benchmark de POSEIDON sur le temps d'exécution

D'après leur recherches, le hachage POSEIDON primitif peut être représenté comme un programme avec peu de registres, un petit nombre d'étapes et un faible degré, bien plus performant que les autres algorithmes de hachage, et plus en accord avec les limitations et la sécurité inhérente du ZKP. Les tableaux précèdent sont un comparatif des temps d'exécution de POSEIDON avec d'autres techniques de hachage.

4.4 Informations de certificat vérifiables par un tiers :

4.4.1 Joseph K. Liu et al. 2021

Dans cet article [18] , Joseph K. Liu et al. Propose un protocole qui permet aux utilisateurs d'être avertis s'ils sont restés contact étroit avec un patient confirmé. Le protocole est conçu pour équilibrer la confidentialité, la sécurité et l'évolutivité. Plus précisément l'application permet à tous les utilisateurs de masquer leurs emplacements passés et l'historique des contacts de gouvernement,

sans affecter leur capacité à déterminer s'ils ont eu un contact étroit avec un patient confirmé dont l'identité ne sera pas révélé.

- Phase d'enregistrement : chaque utilisateur choisit sa clé secrète et sa clé publique, et télécharge la clé publique sur le site du gouvernement pour l'inscription. Cela permet au gouvernement de relier les clés publiques avec le nom ou l'identité de l'utilisateur. Il s'agit de rendre des comptes liés et empêcher l'enregistrement multiple du même utilisateur. Un médecin reçoit une clé secrète individuelle supplémentaire délivrée par son organisation affiliée (par exemple, hôpital), qui est utilisé pour générer une signature de groupe au nom de l'organisation affiliée. Cette étape fait également partie du protocole de défi-réponse.
- Phase de réunion : l'application de chaque utilisateur envoie un package vers les smartphones des autres utilisateurs à intervalles réguliers.À la réception d'un nombre seuil du même colis dans un certain délai, l'application confirmera l'utilisateur concerné comme un cas contact Covid. Les informations d'identification seront utilisées plus tard pour prouver au médecin que l'autre personne est un cas contact.

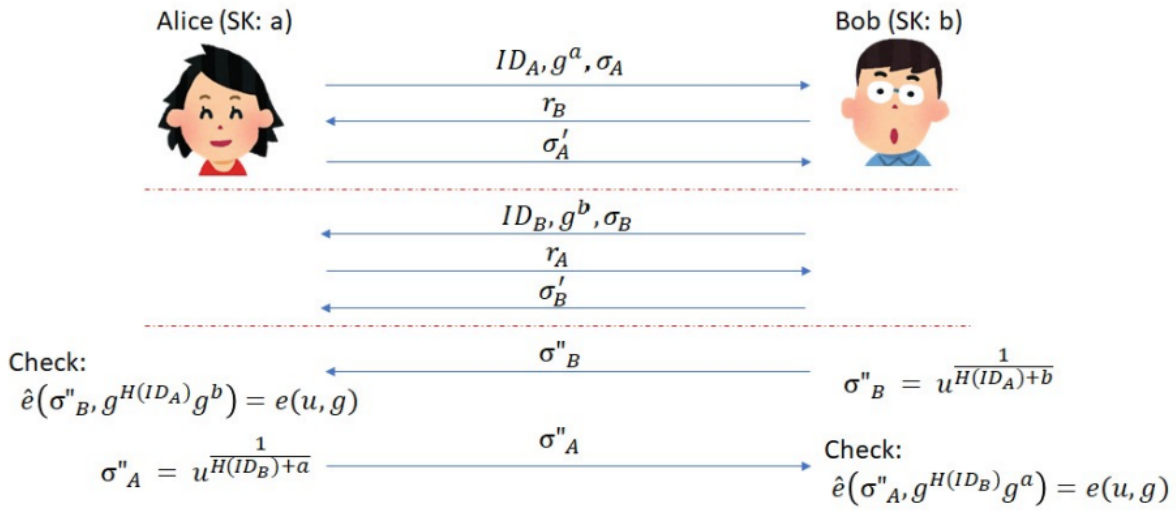


FIG. 4.4 : Schéma de preuve de cas contact ZKP

4.5 Synthèse

Toutes les approches examinées ont réussi à élaborer des utilisations pratiques d'identité en utilisant le ZKP, qui reposent sur le schéma de Shnorr. Bien que chaque méthode offre des avantages en ajoutant des fonctionnalités au protocole de base, certaines lacunes ont également été identifiées. Dans le cas des identités digitales , L'approche de MOHAMEDEN DIEYE et al. en 2016 présente plusieurs avantages notables. Tout d'abord, la confidentialité des partages est garantie par la complexité du protocole décrit précédemment, et l'intégrité de la clé privée est assurée par une fonction de hachage. De plus, l'algorithme ne divulgue aucune information sur la clé secrète. Cette approche ajoute des fonctionnalités importantes a l'identification digitale grâce au ZKP ;Cela diminue le risque de plusieurs types d'attaques potentielles qui pourraient écouter les échanges pour voler des informations confidentielles . Cependant, le hachage utilisé n'est pas spécifique au ZKP, ce qui le rend redondant pour les utilisations d'authentification. Dans l'approche de Lorenzo Grassi et al., un nouveau type de hachage est démontré comme étant plus utiles pour les applications de ZKP.
Dans les systèmes d'identification vérifiables par un tiers, tel que décrit dans la recherche de Joseph K. Liu et al. ,elle permet de préserver la confidentialité des utilisateurs et d'utiliser leurs informations privées pour éviter des cas contact, mais il existe un problème de dépendance , car si le tiers qui fournit le service de vérification d'identité rencontre des pannes ou cesse de fournir ce service, cela peut affecter la capacité du système ZKP à fonctionner correctement. cela peut entraîner la perte des informations d'identification des utilisateurs .

4.6 Conclusion

Dans cet état de l'art, nous avons présenté différentes implémentations comportant une variété de protocoles dans le domaine de partage de secret en utilisant **le Zero Knowledge Proof.**
C'est grâce à cette étude qu'on s'est inspirés à établir notre propre protocole pour développer un **système de vérification des certificats**

Chapitre 5

Contribution

Sommaire

5.1 Introduction

Dans le chapitre précédent nous avons exposé différentes implémentation dans trois différents types de **ZKP** :Informations de certificat vérifiables par un tiers, de Hachage spécifique pour les ZKP et Identité digitale basé sur le ZKP. Pour ce chapitre, nous allons présenter une méthode d'authentification des certificats dans un écosystème décentralisé, puis la partie modélisation . Ensuite nous présentons les démarches d'implémentation suivies avec les résultats obtenus.

Pour cela nous avons besoin d'outils de modélisation et de programmation. Nous nous sommes basés sur :

1. Modelio open source : du côté de modélisation.
2. Java : du côté d'implémentation.
3. JSP Servlets : pour la plateforme web

5.2 Les enjeux

Les informations d'identification vérifiables font partie de la vie quotidienne ; les permis de conduire sont utilisés pour affirmer le droit de conduire, les diplômes universitaires peuvent être utilisés pour affirmer le niveau d'éducation et les passeports délivrés par le gouvernement permettent de voyager.

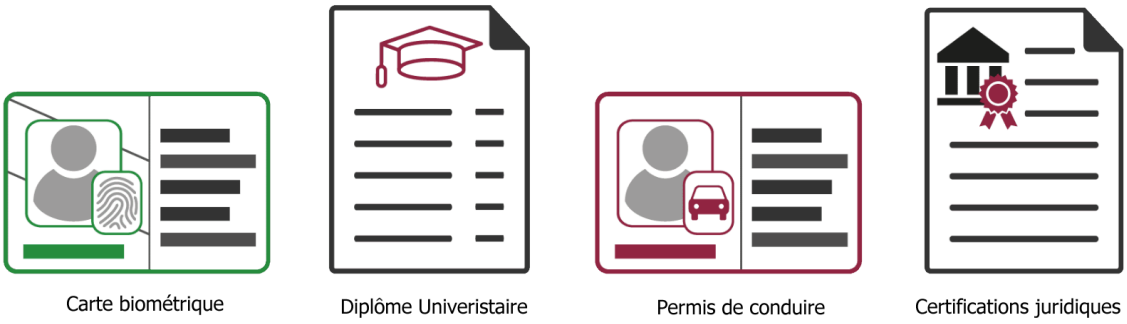


FIG. 5.1 : Différents types d'informations vérifiables

Ces informations d'identification offrent des avantages lorsqu'elles sont utilisées dans le monde physique, mais leur utilisation sur le Web continue d'être difficile.

Actuellement, il est difficile d'exprimer sur le Web les diplômes, les données de santé, les détails des comptes financiers et d'autres types d'informations personnelles vérifiées par des tiers, par leurs sensibilité [19]. La difficulté d'exprimer ces informations d'identification numériques sur le Web rend difficile de recevoir les mêmes avantages via le Web que les informations d'identification physiques nous offrent dans le monde physique.

Heureusement, les protocoles de preuve à divulgation nulle de connaissance (ZKP) offrent un moyen de contrecarrer ce problème, ainsi que d'offrir des fonctionnalités nouvelles et intéressantes

5.3 Approche proposée

Notre environnement choisi est l'écosystème d'identité décentralisée (DID) où les utilisateurs sont chargés de gérer leurs propres identités et informations d'identification

Les identifiants décentralisés (DID) sont un type d'identifiant qui permet une identité numérique décentralisée vérifiable. Ils ont été conçus de manière à pouvoir être dissociés des registres centralisés, des fournisseurs d'identité et des autorités de certification [14].La conception permet au contrôleur d'un DID de prouver le contrôle sur celui-ci sans demander l'autorisation d'une autre partie.

Nous allons développer un nouveau schéma Zero-Knowledge Proof qui améliore l'utilité des identifiants décentralisés (DID) en ajoutant des informations d'identification vérifiables, totalement sécurisés et contrôlés entièrement par l'utilisateur. Cela permettrait dans l'exemple choisi, d prouver des certifications de diplôme a un tiers parti employeur, sans avoir a contacter une académie par exemple.

5.3.1 Cas d'études

L'un des cas d'utilisation initiaux sur lesquels de nombreux écosystèmes d'identité décentralisés se concentrent est l'utilisation de DID pour des informations d'identification vérifiables (VC).
Une informations d'identification vérifiables ou verifiable credential est une affirmation signée du propriétaire du DID (appelé "l'émetteur") à une autre entité (appelée "utilisateur") à qui il certifie un fait ou une preuve. L'utilisateur peut ensuite authentifier les faits contenus en soumettant la preuve à d'autres parties (appelées « vérificateurs ») qui peuvent le vérifier de manière indépendante. Ces informations d'identification peuvent représenter à peu près n'importe quoi, des billets d'avion aux diplômes, mais le concept de base est le même :

- L'émetteur signe une informations d'identification vérifiables pour l'utilisateur à l'aide des clés cryptographiques.
- Le sujet prend possession des informations d'identification signées par l'émetteur et les stocke dans sa Base de Donnée pour une utilisation avec les vérificateurs qui peuvent avoir besoin d'une preuve des faits attestés par l'émetteur.
- Le sujet fournit les informations d'identification signées à un vérificateur qui valide les informations d'identification

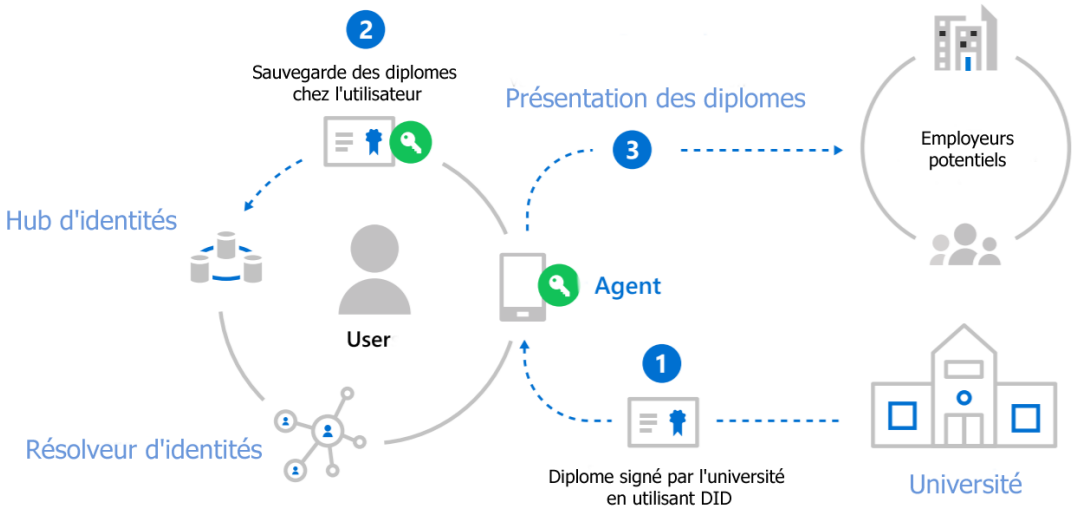


FIG. 5.2 : Modèle d'identité décentralisé

Un tel système pourrait résoudre des cas d'utilisations courants, sur lesquels nous pensons qu'une solution devrait être proposé : La publication de CV en ligne

Pour ce cas d'utilisation, nous avons besoin de vérifier les certifications signées par un émetteur vérifiables par des observateurs .

Ces certifications peuvent être vérifiés dans les cas où il n'y a pas de connectivité. Dans ces cas, nous avons besoin d'un système d'accréditation qui permet aux vérificateurs de mettre en cache des preuves et de les utiliser ultérieurement pour vérifier les informations d'identification localement, sans se connecter à un tiers.

Si la vérification est anonyme, il faut appliquer une protection contre le partage de certificats entre deux utilisateurs, donc la sauvegarde des IDs.

Si l'employeur potentiel est un concurrent de l'employeur actuel de l'utilisateur, cela peut causer des problèmes. De même pour des employeurs qui communiquent entre eux contre un utilisateur. Nous avons donc besoin d'un système de certification qui permette aux parties de vérifier le statut sans révéler aucune information , même en ce concertant ou travaillant ensemble.

5.3.2 Problématiques courantes

Notre schéma parvient a déjouer des contraintes courantes dans les systèmes d'accréditation

1. La présentation des mêmes informations d'identification à plusieurs vérificateurs leur permet de suivre son parcours d'utilisation, même dans un système anonyme.

2. L'exposition des valeurs d'identification aux vérificateurs peut exposer inutilement des informations sensibles. Par exemple : divulguer la date de naissance spécifique au lieu de seulement la preuve qu'il est majeur.

3. Les mécanismes de vérification qui divulguent des informations d'identification du sujet permettent aux émetteurs de suivre la façon dont un sujet utilise une informations d'identification vérifiable.

5.3.3 Fonctionnalités attendues

1. Générer des versions non corrélables d'un identifiant pour chaque vérificateur, sans contacter l'émetteur

2. Autoriser les sujets à ne divulguer que le minimum de faits nécessaires

3. Un utilisateur ne peut pas falsifier les informations d'identification générées par l'émetteur

4. Un groupe de vérificateur ne peut pas lier les activités des utilisateurs à travers leur coopération.

Outils cryptographiques

Le schéma précédent nécessite de produire des preuves de connaissance nulle. Pour cela, nous nous appuyons sur des arguments de connaissance succincts non interactifs à connaissance nulle (zkSNARKs) , un type de preuve de connaissance sans connaissance avec des preuves succinctes et des temps de vérification rapides.

Fonctions de hachage

En s'appuyant sur un système de preuve à usage général tel que zkSNARKs, nous pouvons utiliser des fonctions de hachage standardisées (par exemple, SHA-256) et des schémas de signature numérique (par exemple, ECDSA). Nous avons décider d'utiliser POSEIDON [17], tel que vu dans l'état de l'art.

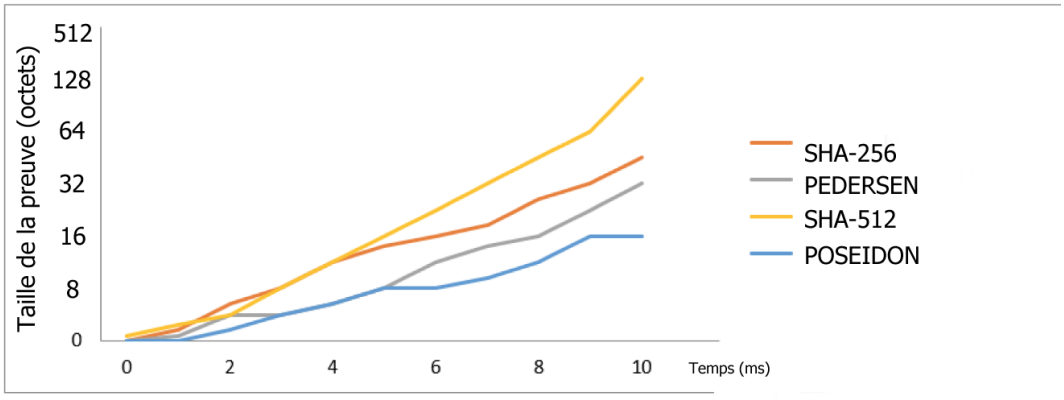


FIG. 5.3 : Benchmark de différents hachages

Nos résultats préliminaires avec POSEIDON (fig 5.3) ont de bons résultats en termes de tailles des preuves hashés, en fonction d'une preuve zkSNARK de taille de 15 Ko, c'est pourquoi nous l'avons choisie.

Bibliothèque et langage

Nous utilisons des bibliothèque Java pour les systèmes distribués de preuve à zéro connaissance.La bibliothèque libsnark fournit un système de preuve zkSNARK distribué qui permet des calculs vérifiables jusqu'à des milliards de portes logiques, dépassant de loin l'échelle des solutions de pointe précédentes. Nous avons trouvé que Java correspondait à nos objectifs car nous pouvions tirer parti de ses riches fonctionnalités pour la programmation orientée objet et nous pouvions contrôler l'exécution de manière fine.

5.4 Notre Solution

Nous avons développé une solution qui repose sur l'utilisation de la preuve de connaissance Zero Knowledge Proof (ZKP) ainsi que des signatures numériques et des fonctions de hachage. Cette combinaison de techniques avancées nous permet d'assurer un niveau élevé de sécurité, tel que décrit dans le chapitre sur les Zero Knwoledge Proof.

5.4.1 Architecture du réseau de partage

L'architecture de base de notre système de partage nécessite trois types d'entités, pas besoin de serveur centralisé ni de base de donnée centralisé, car nous sommes dans un contexte décentralisé. Un client Émetteur , qui offre une certification. Un client Détenteur qui veut démontrer une information certifiée, et enfin un Client Vérifieur qui veut vérifier cette information .

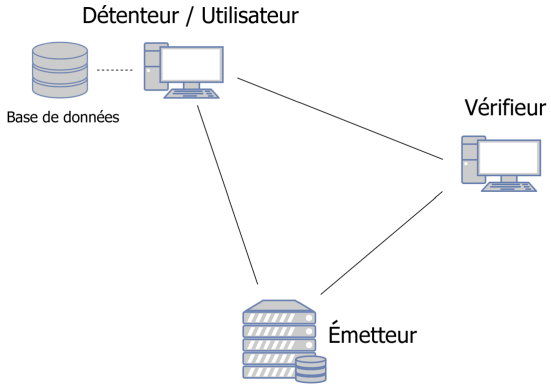


FIG. 5.4 : Architecture Réseau

Symbole	Description
R	nombre aléatoire
salt	nombre aléatoire utilisé pour le salage
DID(U,I)	Identité décentralisée entre émetteur et User
DID(U,RP)	Identité décentralisée entre Vérifier et User
Pk(U,I)	Clé publique de la relation DID(U,I)
I pk	clé publique de émetteur
I sk	Identité décentralisée entre Vérifier et User
Atts	Une liste d'attributs décrivant User
Claim	La liste d'attributs déclaré par User
σ	La signature générée des données atts
H	Une fonction de hachage
h	Une partie de la liste des hachages, envoyée par User
Sign	Une fonction de signature
Verify	Une fonction de vérification de la signature Sign
Pseudonym	Le pseudonyme généré liée a l'utilisateur

TAB. 5.1 : Notations

Phase d'émission des informations d'identification

L'utilisateur s'authentifie auprès de l'émetteur en tant que propriétaire de l'identifiant décentralisé DID(U,I) L'émetteur détient une paire de clés de signature Ipk, Isk qui sont liées au DID de l'émetteur.

Pour émettre une information d'identification à l'utilisateur avec l'identifiant décentralisé DID(U,I)pour les attributs atts, l'émetteur procède comme suit :

Algorithme 5 : Algorithme d'initialisation de la signature

Variables : : Clé publique ; pk **:Identifiant** ; DID(U,I) ;

1 `Début :`

2 r $\longleftarrow random()$;
$salt \longleftarrow random()$;
$\sigma \longleftarrow Signe(Isk, (pk(U,I)atts, r))$;

3 **Retour :** r, salt , σ
Fin ;

L'émetteur maintient un tableau où il y a une ligne distincte pour chaque identifiant actif par DID avec l'information suivante :

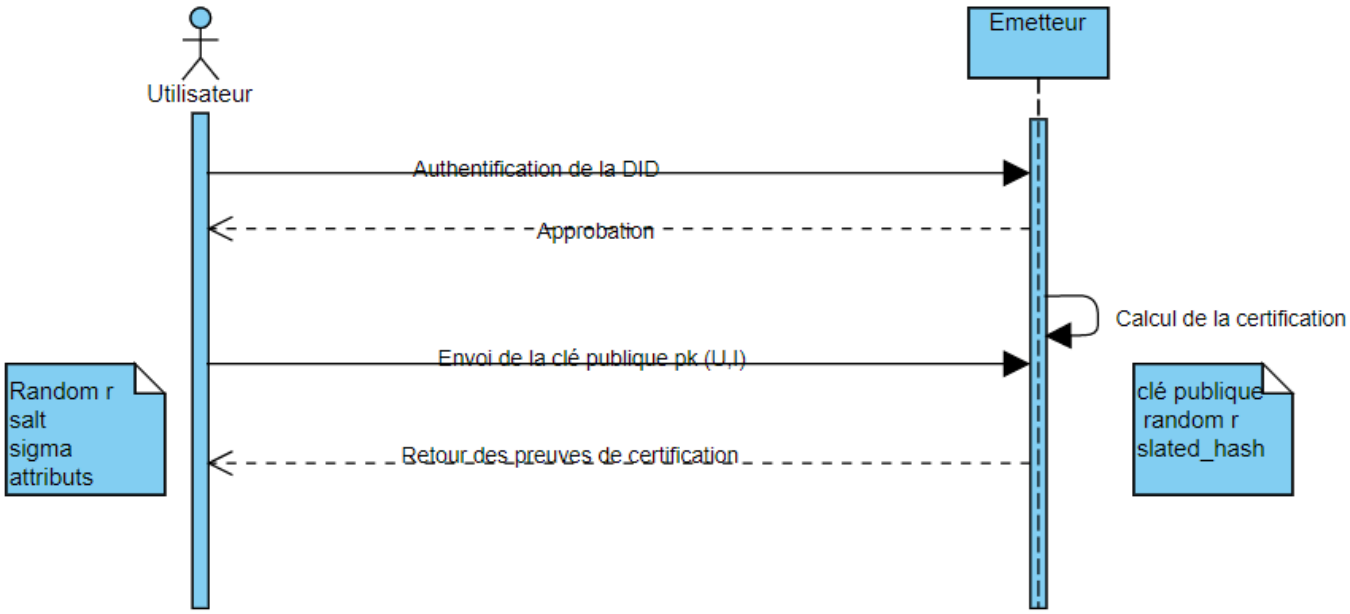


FIG. 5.5 : Phase d'émission des informations d'identification vérifiables

Symbole	Description
R	nombre aléatoire
DID(U,I)	Identité décentralisée entre émetteur et User
Pk(U,I)	Clé publique de la relation DID(U,I)
H(salt, ats)	Le hachage et salage des attributs

TAB. 5.2 : Base de données coté émetteur

L'utilisateur stocke :

Symbole	Description
R	nombre aléatoire
Atts	Attributs liées au certificat
Salt	Le sel utilisé pour le salage

TAB. 5.3 : Base de données coté Utilisateur

Phase de Révocation des identifiants

Lorsqu'un identifiant est révoqué, l'émetteur supprime l'entrée pour cet identifiant de sa table. Cela signifie que l'identifiant révoqué ne sera plus reconnu ou accepté par l'émetteur lors des prochaines communications ou interactions.
L'émetteur doit prendre des mesures appropriées pour s'assurer que la révocation des identifiants est correctement traitée et propagée à tous les systèmes et parties concernées. Cela inclut de renouveler le tableau des identifiants qui sera demandé par le vérificateur ultérieurement.

A noter que la révocation des IDs se fait aussi du coté de l'utilisateur, donc un challenge sur les données utilisateurs est nécessaire.

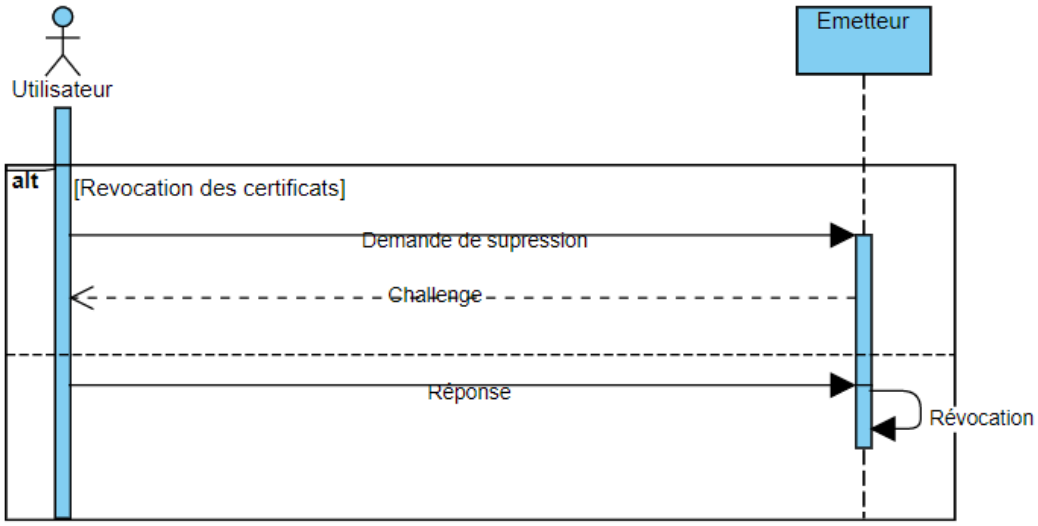


Fig. 5.6 : Phase de révocation des identifiants

Phase de délivrance et récupération des identifiants

Un émetteur signe numériquement les identifiants et la liste de leurs attributs avec sa clé privée.

Le verifieur récupérant la dernière liste d'utilisateurs actifs Le vérifieur s'authentifie auprès de l'émetteur en tant que propriétaire de l'identifiant décentralisé DIDRP.

N'importe quel vérifieur peut télécharger un hachage cryptographique des attributs de chaque identifiant actif, mais afin d'obtenir la non-liaison , ces tableaux sont spécifiques à chaque vérifieur, et pour empêcher la l'apprentissage d'attributs cachés via une attaque par force brute, nous incluons un identifiant aléatoire spécifique nonce r connu uniquement de l'utilisateur et de l'émetteur. Le vérifieur récupérera alors un ensemble de valeurs H(DIDRP, pk(U,I) ,r, H(sel, atts)) pour toutes les entrées du tableau des émetteurs.

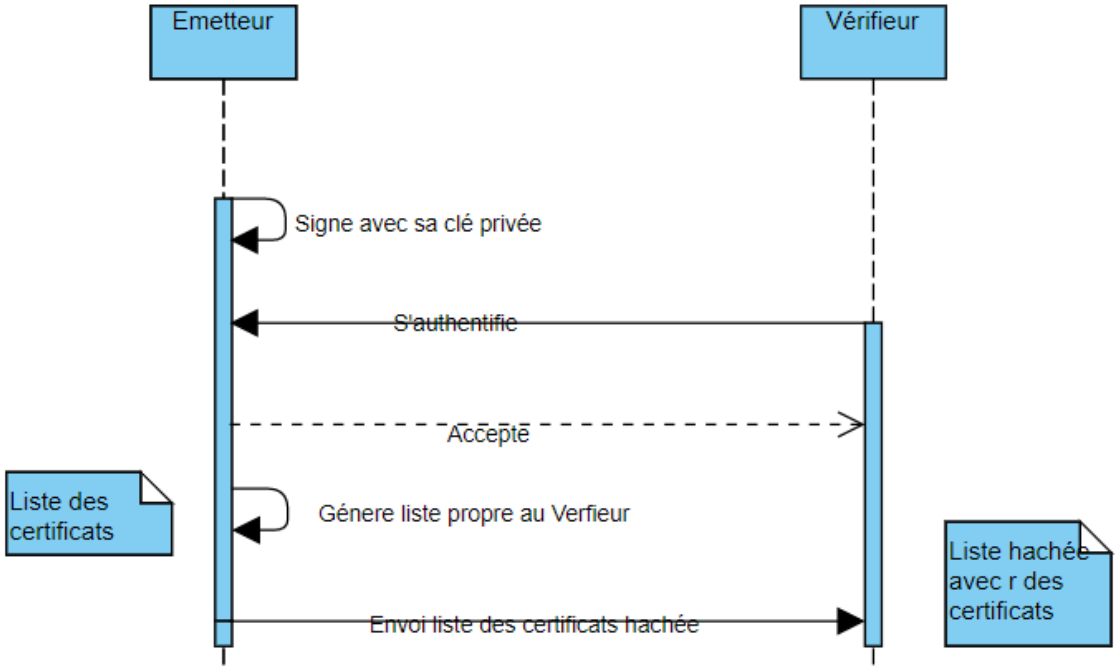


FIG. 5.7 : diagramme de cas d'utilisation.

Phase de présentation des informations d'identification

L'utilisateur s'authentifie auprès du vérifieur en tant que propriétaire de l'identifiant DID(U,RP). Pour qu'un utilisateur puisse prouver les revendications concernant les informations d'identification à un vérifieur, l'utilisateur agit en tant que prouveur dans un système de preuve à connaissance nulle et le vérifieur agit en tant que vérificateur. l'utilisateur produit une preuve à connaissance nulle de la connaissance de c et nonce r et les envoie au vérifieur.
Ce processus de présentation des informations d'identification permet à l'utilisateur de prouver son identité de manière sécurisée et sans divulguer les détails spécifiques de ses informations d'identification. Cela offre une garantie supplémentaire qui garantit la divulgation nulle de connaissance, et l'impossibilité de lier deux fois le même utilisateur qui viendrait a s'authentifier, donc anonymement.

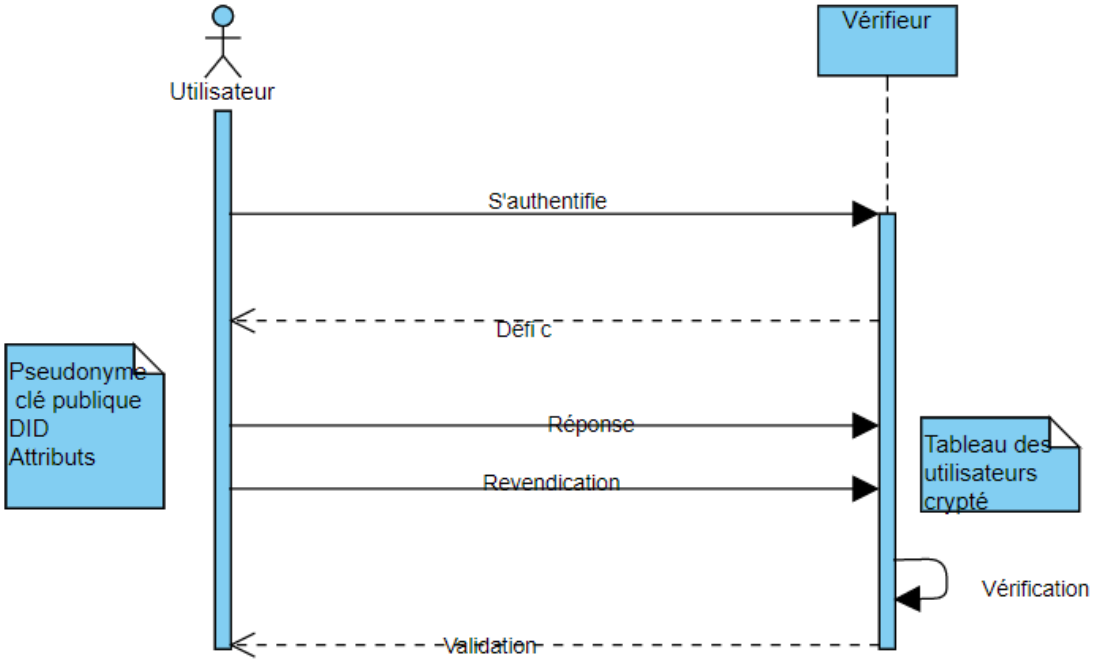


FIG. 5.8 : Modèle d'identité décentralisé

Phase de Vérification des informations d'identification

Le vérifieur vérifie la preuve et vérifie si elle est présente dans sa liste d'utilisateurs actifs extraite de l'émetteur.

Pour cela , nous avons deux algorithmes possibles :

- Prouver la connaissance d'une signature σ, une paire de clés pk(u, i) , sk(u, i) , un nonce r, et un ensemble d'attributs atts. Ce premier algorithme assure l'anonymat, mais ne permet pas de savoir si une information a été révoquée auparavant.

Algorithme 6 : Algorithme de vérification "1"

Variables : : Clé publique ; Ipk **: Clé publique du DID(U,I)** ; pk(U,I) **: Attributs** ; atts **:Nonce** ; r ;

```
Début :
si Verify(Ipk,(pk(U,I),atts,r)) == accept ; alors
    si Paire(pk,sk)==valide alors
        Validation ⟵ Vrai
sinon
    Validation ⟵ Faux
si claim==atts ; alors
    Revendication ⟵ Vrai
sinon
    Revendication ⟵ Faux
Retour : Validation , Revendication
Fin ;
```

- Prouver la connaissance d'un hachage d'attributs du tableau , une paire de clés pk(u, i) , sk(u, i) , un nonce r, et un ensemble d'attributs atts . Cet algorithme permet de s'assurer que l'identifiant n'a pas été révoquée au prix d'une protection contre la réutilisation des identifiants, donc par le stockage d'un pseudonyme caractérisant l'utilisateur

Algorithme 7 : Algorithme de vérification "2"

Variables : : Clé publique ; Ipk **: Clé publique du DID(U,I)** ; pk(U,I) **: Attributs** ; atts **: Pseudonyme** ; pseudonym

:Nonce ; r ;

```
1  Début :
2  si H(DIDRP, pk(U,I),r, H(salt, atts))=h ; alors
3      si Pseudonyme == H(DIDRP,r) AND Paire(pk,sk)==valide alors
4          Validation ⟵ Vrai
5      sinon
6          Validation ⟵ Faux
7  si claim==atts ; alors
8      Revendication ⟵ Vrai
9      sinon
10     Revendication ⟵ Faux
11 Retour : Validation , Revendication
   Fin ;
```

Dans ce cas, le vérifieur vérifiera que h figure dans la liste des hachages qu'il a obtenus de l'émetteur, et ce pseudonyme n'a jamais été vu auparavant avec un DID différent.

5.4.2 Temps d'exécution

Dans les zkSNARK, le coût de production d'une preuve est important à prendre en compte.

Nous estimons les coûts de notre système pour prouver des prédicats simples sur un justificatif d'identité avec 15 Ko (attributs codés) signé à l'aide de POSEIDON.

Dans notre implémentation, sur un seul coeur de processeur d'un ordinateur Microsoft Surface 8, prend moins de 170 ms pour produire une preuve d'une taille de 15 Ko qui peut être vérifiée en 17 ms.

5.5 Sécurité

Ce système atteint les propriétés d'un Zero Knowledge Proof tels que décrit dans les chapitres suivants :

- Aucun apport d'information : Parce que le r est aléatoire et inconnu de l'un des vérifieurs, nous avons la garantie que

  h1 = H(DIDRP1, pk(U,I) ,r, H(sel, atts)), pseudonyme1 = H(DIDRP1,r)

et h2 = H(DIDRP2, pk(U,I) ,r, H(sel, atts)), pseudonyme2 = H(DIDRP2,r)

ressemblent à 4 chaînes aléatoires du point de vue des vérifieur 1 et 2, même s'ils mettent en commun tous leurs information.Ils ne seront pas en mesure de dire que le même utilisateur a présenté un identifiant aux deux vérifieur. Ainsi, la seule chose que le vérifieur apprend est que l'information d'identification est vraie.

- Consistance : Seul un utilisateur ayant l'accès a toutes ces informations , et ayant déjà été certifiée pourra déclarer des informations et être correctement approuvé.
- Robustesse : Le pseudonyme est calculé à partir du r, qui est fixe pour un identifiant donné. Donc même un utilisateur malveillant ayant accès au identifiants d'un utilisateur , sera détecté.

5.6 Étude comparative

Il y a un intérêt croissant pour la construction d'un écosystème d'identité décentralisée (DID), notamment grâce à un identifiant basé sur les revendications , qui vise à offrir de fortes garanties de confidentialité. Nous reprenons les études discutées dans les précédents chapitres et voyons comment notre proposition se compare à eux. Toutes les propositions utilisent un modèle similaire et spécifient entre trois types d'entités :(1) un émetteur, (2) un utilisateur et (3) un Vérifieur (RP).

1. BZDIMS. La principale différence entre notre proposition et celle de BZDIMS réside dans la révocation des identifiants.

   Dans notre protocole, les vérifieur peuvent télécharger la dernière liste des utilisateurs actifs des émetteurs et vérifier le statut de révocation de toutes les revendications.

2. Identifiants anonymes. Les informations d'identification anonymes, permettent aux utilisateurs d'obtenir des informations d'identification des émetteurs, puis les présenter aux vérifieur, en ne révélant que si leurs attributs satisfont aux revendications données. Il ne permettent pas de dissocier plusieurs présentations du même titre

3. Signatures locales du vérificateur permettent des contrôles de révocation différés, mais fournissent une garantie plus faible en ce sens qu'une fois qu'un identifiant est révoqué, toutes les utilisations précédentes et futures de ce justificatif d'identité peuvent être liées

Approche	Révocation des certificats	Divulgation nulle	Dissociation
BZDIMS		X	X
Identifiants anonymes	X	X	
Signatures locales	X	X	
Notre Approche	X	X	X

Tab. 5.4 : Tableau comparatif des approches

5.7 Présentation de l'application

Pour simuler le protocole que nous avons proposé, nous avons entrepris le développement d'une application web avec une interface utilisateur et une interface vérificateur distinctes, le tout simulé sur deux machines virtuelles différentes, et une communication grace au Sockets Java.

Dans ce qui suit, nous présentons les différentes interfaces et avons inclus l'exécution côté serveur en utilisant Java, dans le but d'améliorer la lisibilité des résultats.

Il est important de noter que nous avons délibérément inclus les données obtenues ,les IDs, les résultats de hachage, et autre attributs, lors de cette version de simulation afin de mettre en évidence la complexité des calculs effectués.

5.7.1 Phase d'authentification de l'utilisateur

L'application se présente tel que dans la figure 5.9. L'utilisateur s'authentifie en cliquant sur le bouton au milieu.

Cela enclenche la phase 1, la phase d'émission des informations d'identifications entre l'utilisateur et l'émetteur. Le détenteur utilise sa fonction de hachage sécurisée pour convertir l'identifiant décentralisé DID(U,I) en une représentation sous forme de hachage et ajoute des mesures de sécurité supplémentaires avec le salage. L'ensemble des attributs et alors envoyés vers l'utilisateur, que l'on a choisi de montrer dans l'interface dans la fig 5.10

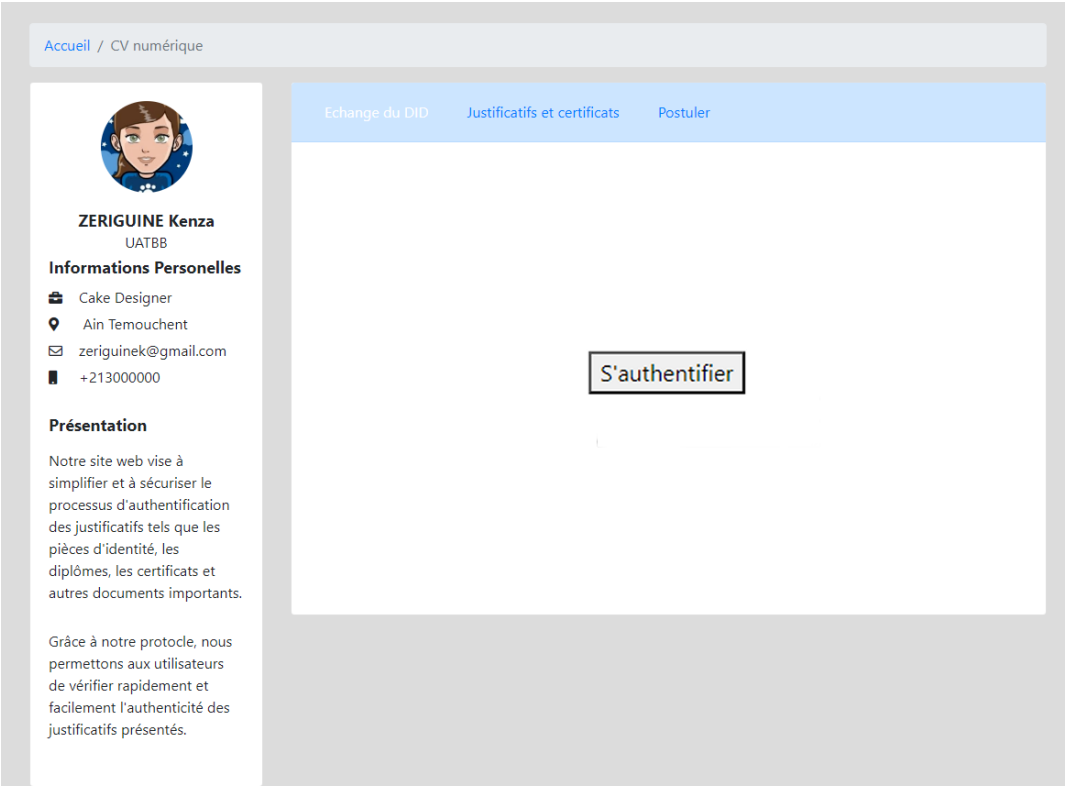


FIG. 5.9 : Authentification de l'utilisateur

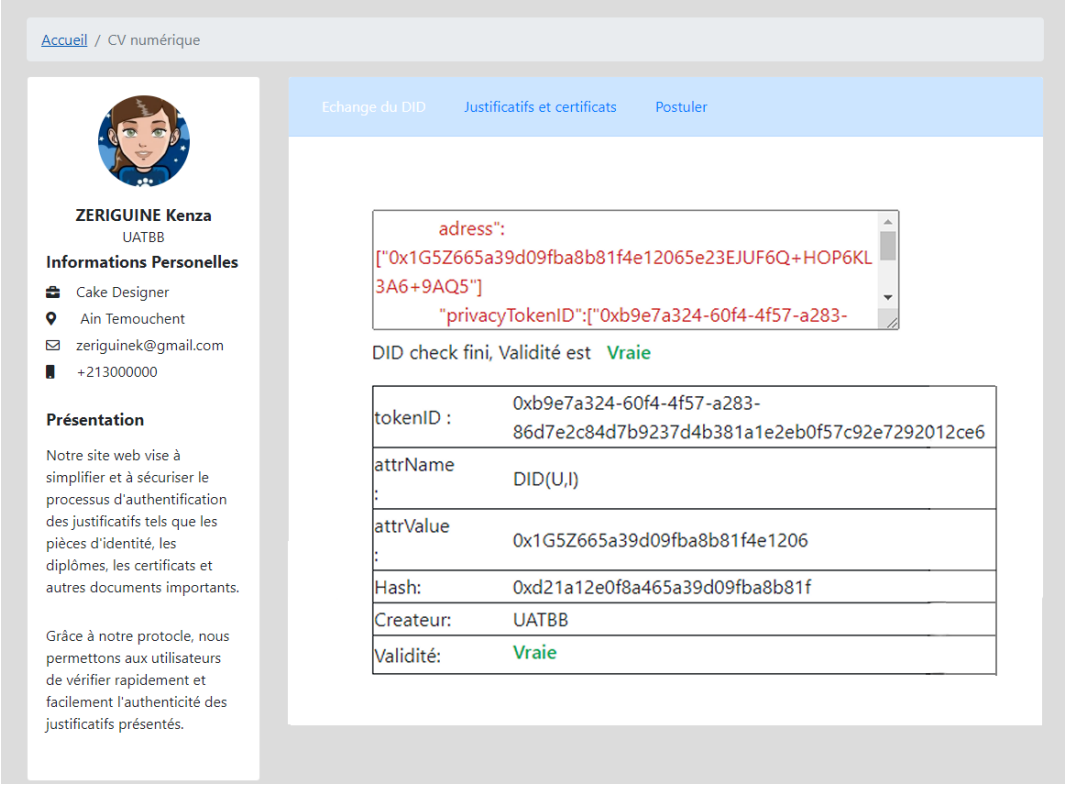


FIG. 5.10 : Phase d'émission des informations d'identifications

5.7.2 Phase de délivrance

L'utilisateur peut demander une nouvelle certification en allant dans la rubrique Justificatifs et certificats. Dans la fig 5.11 ,on peut voir les certificats déjà établit dans la base de donnée de l'utilisateur. En cliquant sur le bouton "+", cela enclenche la phase 2 , la délivrance et récupération , tel que décrite dans la section précédente dans la fig 5.7

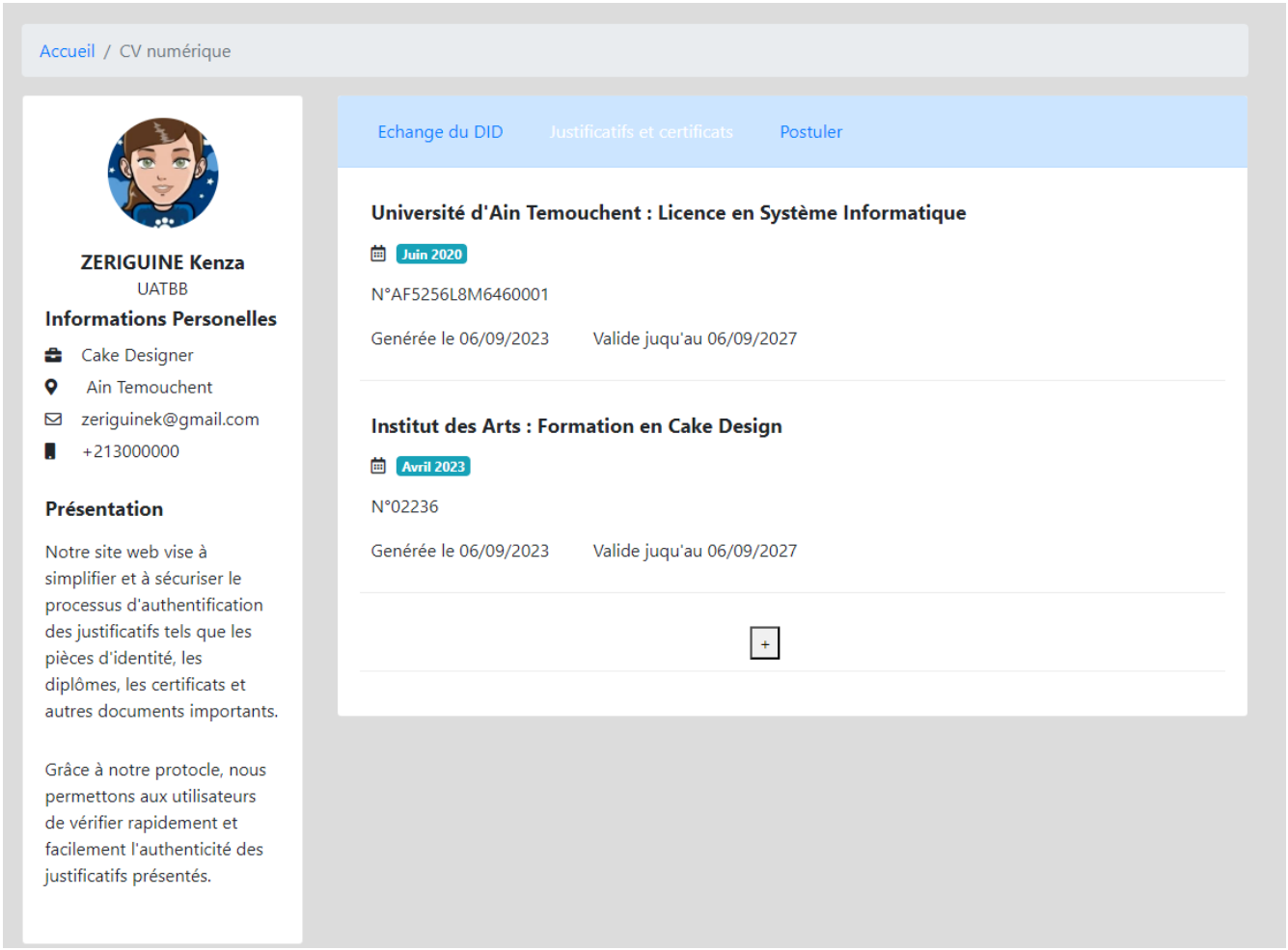


FIG. 5.11 : Certifications de l'utilisateur

Le fichier JSON de sortie est un fichier JSON-LD, conforme au normes W3C des certifications vérifiables (verifiable credentials). Le fichier JSON-LD de sortie comprend une liste des certifications vérifiables, contenant des attributs tels que le nom du titulaire, l'émetteur de la certification, la date d'émission, la date d'expiration et d'autres informations pertinentes. Chaque certification est également associée à une signature numérique et un nombre random, tel que décrit dans le tableau de la base de données fig 5.3.

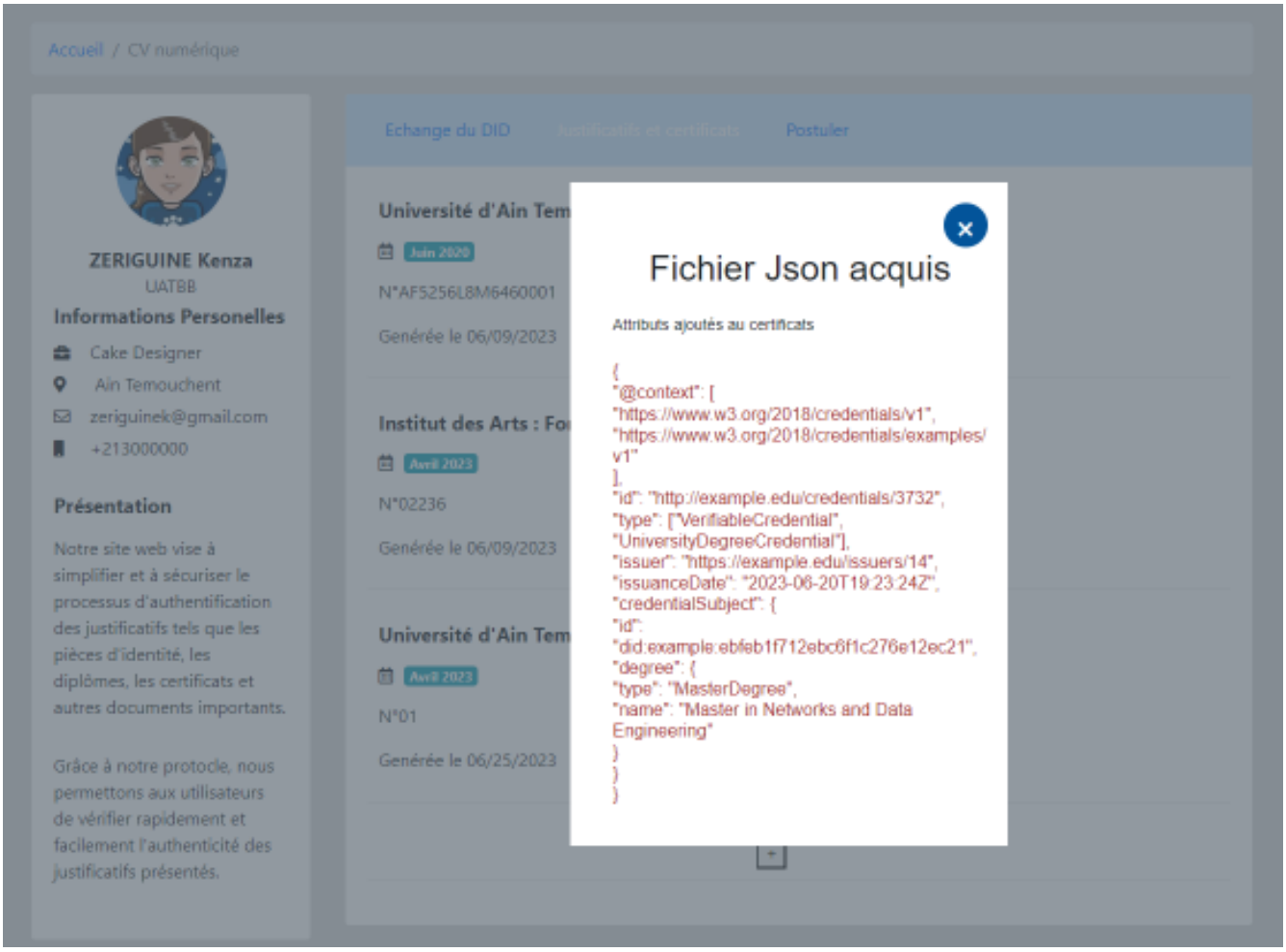


Fig. 5.12 : Acquisition du fichier JSON

Il n'est pas nécessaire de le cacher car n'importe quel utilisateur peut prétendre a ces informations particulières, mais seul un utilisateur précédemment authentifier auprès de l'émetteur pourra les faire vérifier ultérieurement.

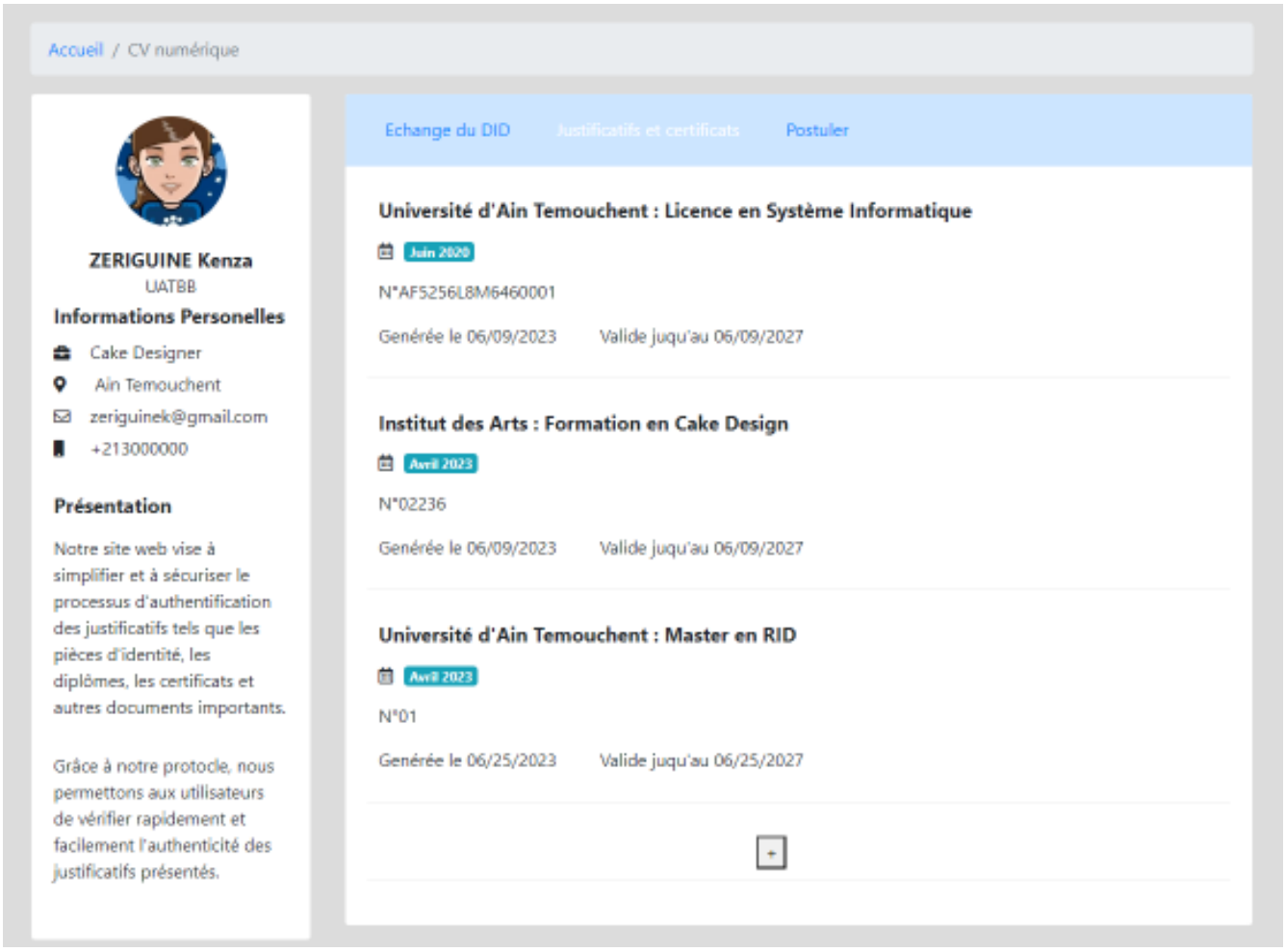


FIG. 5.13 : Nouvelle certification ajoutée

La communication entre l'utilisateur et l'émetteur n'étant plus utile, nous pouvons sortir de l'exécution serveur.

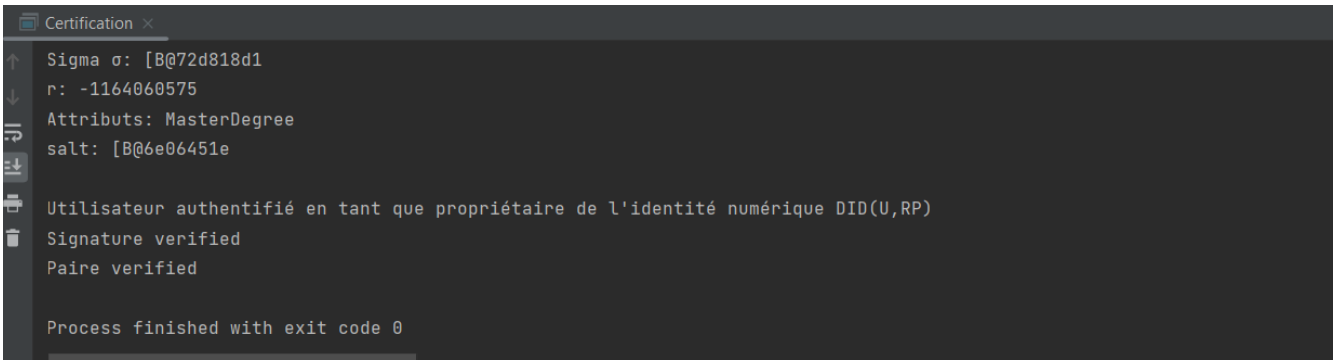


FIG. 5.14 : Exécution du serveur Java

5.7.3 Phase de présentation

Dans notre exemple , un utilisateur vient de postuler chez le vérificateur. Ce dernier n'a accès ni a son nom, ni son email, mais peut vérifier ses justificatifs après avoir télecharger la liste des attributs hashés de l'émetteur.

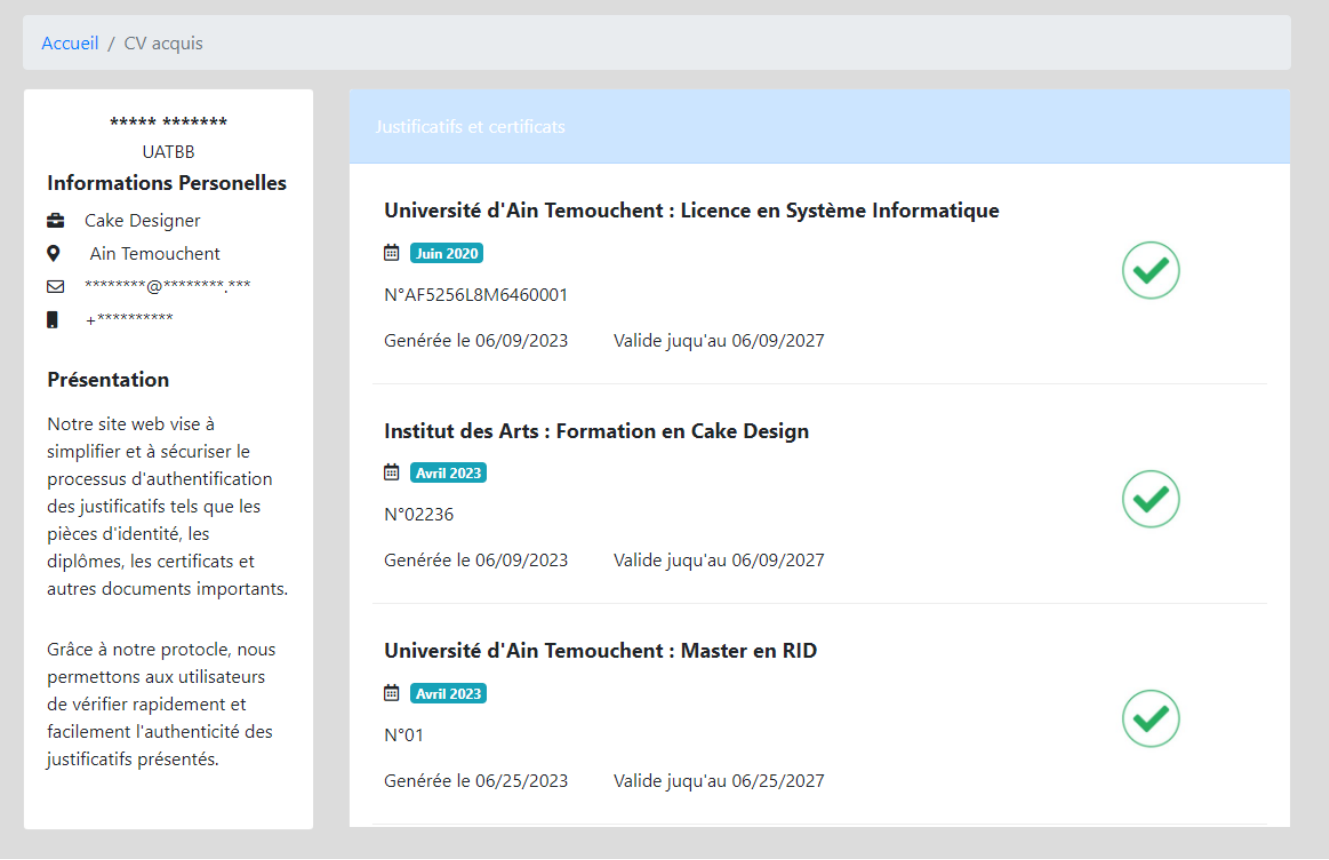


Fig. 5.15 : Certifications authentifiées

5.8 Conclusion

Dans ce chapitre, nous avons présenté notre contribution proposée pour un système d'accréditation basé sur le Zero Knowledge Proof. Nous avons exposé les formules théoriques ainsi que les différentes étapes et outils nécessaires pour les mettre en œuvre. De plus, nous avons effectué une analyse des résultats de quelques tests afin de démontrer sa fiabilité et performance comparative.

L'objectif d'anonymisation des justificatifs et certificats dans un système décentralisé ayant été atteint , nous sommes en mesure de clore avec de bons résultats et une interface prête a être déployée.

Chapitre 6

Conclusion Générale

En cryptographie , Les ZKP sont une manière de prouver la connaissance d'une information sans rien dévoiler sur ce secret, et permettent d'augmenter la sécurité d'un cryptosysteme.

Lors de ce mémoire , nous avons atteint l'objectif donné qui était de mettre en place un système d'identification et d'authentification en utilisant le Zero Knowledge Proof, et construire un protocole de certification basé sur le ZKP qui respecte la vie privée et les données des utilisateurs.

Après avoir implémenter les différentes opérations étudiées on a pu maîtriser dans les moindres détailles les théories abordées Notre étude théorique a été très productive en se basant sur différents livres et articles, cependant les implémentations et les études sur le Zero Knowledge Propof en tant que protocole de partage de certificat vérifiable sont limitées et mal documentée, malgré son potentiel d'utilisation dans le domaine.

Dans la phase pratique , des registres numériques distribués auraient été plus pratiques afin de tester correctement notre protocole . Pendant notre études de différents articles nous avons trouver des pistes de futurs implémentations des informations vérifiables grâce au Zero Knowledge Proofs en utilisant la blockchain ce qui permettrait en théorie de facilité la transition vers un Web3 totalement décentralisé et sécurisé,

Nous espérons bien au futur pouvoir améliorer le protocole afin de permettre une vérification plus linéaire et plus intuitive, ainsi que de résoudre les failles de sécurité relevées dans le chapitre précedent.

Bibliographie

[1] Dobromir Todorov. *Mechanics of user identification and authentication : Fundamentals of identity management*. CRC Press, 2007.

[2] Shafi Goldwasser, Silvio Micali, and Chales Rackoff. The knowledge complexity of interactive proof-systems. In *Providing Sound Foundations for Cryptography : On the Work of Shafi Goldwasser and Silvio Micali*, pages 203–225. 2019.

[3] Nicolas Petit. Are'fangs' monopolies ? a theory of competition under uncertainty. *A Theory of Competition under Uncertainty (October 10, 2019)*, 2019.

[4] Xiaoqiang Sun, F Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE network*, 35(4) :198–205, 2021.

[5] Gerhard J Woeginger. Exact algorithms for np-hard problems : A survey. In *Combinatorial Optimization—Eureka, You Shrink ! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, pages 185–207. Springer, 2003.

[6] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In *Conference on the Theory and Application of Cryptology*, pages 628–631. Springer, 1989.

[7] Kevin S McCurley. The discrete logarithm problem. In *Proc. of Symp. in Applied Math*, volume 42, pages 49–74. USA, 1990.

[8] Ran Canetti, John Friedlander, Sergei Konyagin, Michael Larsen, Daniel Lieman, and Igor Shparlinski. On the statistical properties of diffie-hellman distributions. *Israel Journal of Mathematics*, 120:23–46, 2000.

[9] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Providing Sound Foundations for Cryptography : On the Work of Shafi Goldwasser and Silvio Micali*, pages 329–349. 2019.

[10] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pages 263–280. Springer, 2012.

[11] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *GitHub : San Francisco, CA, USA*, 4:220, 2016.

[12] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for schnorr signatures. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings, Part I 20*, pages 512–531. Springer, 2014.

[13] Shafi Goldwasser and Yael Tauman Kalai. On the (in) security of the fiat-shamir paradigm. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 102–113. IEEE, 2003.

[14] Uwe Der, Stefan Jähnichen, and Jan Sürmeli. Self-sovereign identity − opportunities and challenges for the digital revolution. *arXiv preprint arXiv :1712.01767*, 2017.

[15] Mohameden Dieye, Pierre Valiorgue, Jean-Patrick Gelas, El-Hacen Diallo, Parisa Ghodous, Frédérique Biennier, and Éric Peyrol. A self-sovereign identity based on zero-knowledge proof and blockchain. *IEEE Access*, 2023.

[16] Xiaohui Yang and Wenjie Li. A zero-knowledge-proof-based digital identity management scheme in blockchain. *Computers & Security*, 99:102050, 2020.

[17] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon : A new hash function for zero-knowledge proof systems. In *USENIX Security Symposium*, volume 2021, 2021.

[18] Joseph K Liu, Man Ho Au, Tsz Hon Yuen, Cong Zuo, Jiawei Wang, Amin Sakzad, Xiapu Luo, Li Li, and Kim-Kwang Raymond Choo. Privacy-preserving covid-19 contact tracing app : a zero-knowledge proof approach. *Cryptology ePrint Archive*, 2020.

[19] Johannes Sedlmeir, Reilly Smethurst, Alexander Rieger, and Gilbert Fridgen. Digital identities and verifiable credentials. *Business & Information Systems Engineering*, 63(5) :603–613, 2021.