TRUSTED REMITTANCE SERVICES IN DECENTRALIZED LEDGER TECHNOLOGY USING A THREE-LAYERED STACK: A PROPOSED MODEL

by

Thomas MAKETA LUTETE

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, May 22, 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Professor, Alain April, Thesis Supervisor
Department of Software Engineering and IT at École de technologie supérieure

Professor Kaiwen Zhang, Thesis Co-supervisor
Department of Software Engineering and IT at École de technologie supérieure

Professor Mohamed Cherriet President of the Board of Examiners
Department of System Engineering at École de technologie supérieure

Professor Alain Abran, Member of the jury
Department of Software Engineering and IT at École de technologie supérieure

Mrs. Cherifa Mansoura Liamani, external evaluator
Business Architect at TEK systems

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

APRIL 29, 2020

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

DÉDICACE

À mes parents feux, professeur Thomas Maketa Lutete Senior, et Céline Nkuizulu Mufuta. Papa, j'aurais tant voulu que tu voies ton fils avoir son doctorat et marcher dans tes pas. J'espère que de là ou tu es, tu es fier de moi. J'ai finalement accompli ton souhait, j'ai terminé mes études doctorales.

À mes sœurs, la professeure médecin Vivi Maketa Tevuzula et la Dr. Aurelie Maketa Mukoko. Chères sœurs, depuis tout petit nous avons grandi, et partagé les bons et mauvais moments ensemble. Vous avez énormément contribué à la réalisation de ce projet. Rien que le fait de vous voir terminer vos études a été une grande motivation pour moi.

À ma femme, Laura Maketa Kimbala. À mes filles Céline, Lara, Vivi et Winnie Maketa. Ma femme, tu as été de tous les combats à mes côtés, tu étais seule avec les enfants, pendant que j'étais plongé dans mes recherches. Ces lignes veulent reconnaitre tes sacrifices et ceux des enfants pendant ces moments compliqués. Sans vous je ne pense pas que j'aurais eu la motivation d'aller au bout de cette aventure. À vous, mes enfants, j'espère que je vous inspire à vous dépasser comme mon père m'a inspiré.

À vous tous, les miens, ce document vous est dédié.

REMERCIEMENTS

Je remercie mes superviseurs les professeurs Alain April et Kaiwen Zhang. Professeur April, sans vous je n'aurais jamais fait mon doctorat. Du premier jour ou vous m'avez accepté comme étudiant, jusqu'à la rédaction de cette thèse vous avez toujours été présent. Vous m'avez aidé financièrement, moralement et académiquement. Je vous dois énormément, ces quelques lignes ne pourront pas exprimer tout ce que vous avez fait pour moi et ma famille au cours de ces années. Du fond de mon cœur je vous remercie.

Professeur Kaiwen, merci pour votre temps, vos conseils et votre présence. Grace à vous ce document a pu bénéficier de conseil et de la rigueur scientifique.

SERVICES D'ENVOI DE FONDS AVEC CONFIANCE À L'AIDE DE LA TECHNOLOGIE DE REGISTRE DÉCENTRALISÉ UTILISANT UN MODÈLE À 3 COUCHES: UN MODÈLE PROPOSÉ

Thomas MAKETA LUTETE

RÉSUMÉ

L'envoi de fonds est un service utilisé mondialement pour transférer de l'argent à la famille et aux amis. Dans le monde entier, de nombreuses entreprises proposent des services d'envoi de fonds aux familles qui en dépendent pour leur subsistance. Malheureusement, un élément de la transaction actuelle d'envoi de fonds qui n'est pas suffisamment pris en compte dans le modèle actuel est qu'une majorité des envois sont initiés avec l'intention claire que le destinataire acquiert un service, par exemple, en payant ses frais de scolarité, es frais médicaux, ses frais de subsistances ou d'épicerie avec les montants reçus. Le modèle actuel termine la gestion de la transaction lorsque le bénéficiaire collecte l'argent et ne garantit pas que ce montant sera correctement utilisé pour acquérir le service souhaité par l'expéditeur. Cette recherche étudie cette problématique en proposant un nouveau modèle d'envoi de fonds qui tient compte de toutes les étapes nécessaires pour que les fonds envoyés soient effectivement versés afin de garantir que l'intention initiale de l'expéditeur soit réalisée. Dans ce modèle, le montant envoyé par virement ne sera utilisé que pour le but auquel il a été initialement destiné, comme prévu par l'expéditeur. Dans le cas contraire, l'expéditeur récupèrera son argent.

Ce type d'envoi de fonds est appelé l'envoi de fonds avec confiance en comparaison avec le simple envoi de fonds qui se termine à la suite de la collecte de l'argent par le bénéficiaire. À la suite d'une analyse approfondie des limites actuelles du modèle d'envoi de fonds, en prenant comme exemple la ville de Kinshasa, en République démocratique du Congo, cette recherche propose un modèle générique pour l'envoi de fonds avec confiance. Ce modèle novateur, possède des caractéristiques permettant l'utilisation et la réutilisation de technologiques émergentes nécessaires pour effectuer en toute sécurité une remise de service sécurisé à l'aide d'un modèle logique d'implantation à 3 couches d'abstraction. Ce modèle d'abstraction, indépendant de toute technologie, permet de mieux concevoir et mettre en œuvre, non seulement les transactions d'envoi de fonds avec confiance, mais également d'autres transactions commerciales se comportant de la même manière.

Le modèle d'abstraction à 3 couches est ensuite implanté dans un prototype expérimental réutilisant des technologies émergentes de la technologie des registres décentralisés (DLT) et de la chaine de blocs (blockchain) qui réduisent considérablement l'effort de conception et de développement nécessaire pour implanter le modèle en prototype logiciel expérimental. Les technologies DLT fournissent, de manière native, la couche centrale du modèle d'abstraction à 3 couches, cette couche qui vérifie et garantit la sécurité et l'authentification des acteurs et des envois de fonds avec confiance.

Cette thèse présente toutes les étapes de conception à du modèle d'envoi de fonds à l'aide d'une abstraction en 3 couches. Sa conception est effectuée à la suite d'une enquête sur le terrain, et est validé à l'aide du développement d'un prototype expérimental qui fait l'objet de la conception d'un modèle de validation et d'une simulation.

La simulation, démontre qu'en effet, le modèle proposé peut être mis en œuvre à l'aide de la technologie de la chaine de bloc Éthéréum afin de démontrer son potentiel. La simulation a également démontrée que, lors de l'utilisation d'Éthéréum, en tant que couche de mise en œuvre, des éléments supplémentaires inhérents à cette technologie, telle que la notion de gaz doivent être considérées afin que les utilisateurs finaux puissent bénéficier d'une expérience financière semblable à celle à laquelle ils sont habitués de la part des services d'envoi de fonds actuels.

Mots-clés: envoi de fonds, chaîne de blocs, Ethéréum, technologie de registres décentralisée

TRUSTED REMITTANCE SERVICES IN DECENTRALIZED LEDGER TECHNOLOGY USING A 3-LAYERED STACK: A PROPOSED MODEL

Thomas MAKETA LUTETE

ABSTRACT

Remittance is a global service used to transfer money to family and friends. Throughout the world, many companies offer remittance services to families who depend on them for their livelihood. Regrettably, there is an aspect of the remittance transaction that is not adequately addressed in the current remittance model. A majority of remittance transactions are initiated with the clear intention that the receiver will acquire a service, for example, pay for school fees, medical procedures or groceries. Unfortunately, the current model stops managing the remittance transaction when the beneficiary collects the money and does not ensure that the remittance is correctly used to acquire the intended service. This research addresses this limitation in proposing a model of remittance that considers all the steps necessary to ensure that the remittance transaction concurs with the intention of the expeditor: sending the money, acquiring a service and paying for a service or returning the money in the case that the service is not provided. In this model, the amount transferred through remittance will only be used for its intended purpose as expected by the remittance's expeditor; otherwise, the expeditor of the remittance gets the money back.

This type of remittance of service is called a trusted remittance of service (TRS) compared with the legacy remittance of money that stops with the collection of the money by the beneficiary. After conducting an in-depth analysis of the current limitations of the remittance of money in the city of Kinshasa, in the Democratic Republic of Congo, as an example, this thesis proposes a generic model for TRS. The proposed 3-layered stack model includes the logical (business requirements) and technological requirements needed to safely conduct a trusted remittance of service. This proposed model, independent of any technology, allows for better design and implementation of not only trusted remittance of service transactions, but also of other business transactions that behave similarly.

The validation steps of the proposed 3-layered stack model showed that the emerging distributed ledger technology (DLT) and blockchain technologies reduce the development time needed to develop a prototype. By design, DLT provides the foundational layer of the 3-layered stack model, which is the layer that verifies and further guarantees the security and authentication of the remittance actors and their respective transaction.

This thesis presents all the steps leading to the experimentation of a working prototype based on the proposed 3-layered stack model. The research results concluded that indeed, the 3-layered stack model can be implemented using Ethereum blockchain technology to show the potential of the TRS model proposed. However, it also showed that when using this technology as the foundational layer for implementing the 3-layered stack, additional elements

inherent to this technology, such as the notion of gas or a penalty for running code in the Ethereum Virtual Machine, have to be considered in order to provide end users with a financially seamless experience similar to those they can expect from the current remittance methods of money providers.

Keywords: Remittance, Blockchain, Ethereum, Distributed Ledger Technology, DLT.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

BFT ..... Byzantine Fault Tolerance problem
DCS ..... Decentralization, Consistency and Scalability
DLT ..... Distributed Ledger Technology
DRC ..... Democratic Republic of Congo
EVM .....Ethereum Virtual Machine
MTO ..... Money Transfer Organizations
RA .....Remittance system application
RM ..... Remittance of money
TCR ..... Total Cost of Receiving Payment
TRS ..... Trusted Remittance of Service

INTRODUCTION

The subject of this thesis is inspired by research regarding remittances to Kinshasa, the capital of the Democratic Republic of Congo (DRC). One of the findings of this research is that most of the remittance money sent from abroad to Congolese beneficiaries is sent with a specific intention in the mind of the expeditor, and that intention is to provide a service to the beneficiary, either to pay for school fees, for medical expenses, groceries, or rent, for example.

In the current remittance model applied by popular money transfer organizations (MTO), we found that although the expeditor clearly expresses the intended use for the money transfer, unfortunately, this information is not accounted for. MTOs consider the remittance transaction as completed the moment the beneficiary listed in the transaction's details collects the money. From that moment onward, the expeditor has no control over the money sent. The expeditor must rely on the beneficiary's trustworthiness to utilize the money as it was intended. In this thesis, this model will be called the "**remittance of money" or RM,** in contrast to the model proposed that will be referred to as "**trusted remittance of service" or TRS**.

The thesis is organized as follows: Chapter 1 introduces the research context, objective and methodology; Chapter 2 presents the results of a field research survey performed in the city of Kinshasa, summarizing the remittance problem this research intends to solve; Chapter 3 provides a literature review necessary for understanding the areas of trust, Ethereum blockchain and remittance, and how they link to TRS; Chapter 4 present the steps leading to a proposed trusted 3-layered stack model of a TRS system. It also includes a discussion addressing the risk associated with this proposal and the eight conditions guaranteed by a system that would implement this model; Chapter 5 presents a proof of concept using the Kaiwen DLT decision tree to identify the DLT that will be implemented in the experimental prototype to validate the model proposal; Chapter 6 continues with the design of the prototype TRS software ready for experimentation; Chapter 7 presents the experimental results of the proof of concept and lastly, the final conclusions and future work are discussed.

CHAPTER 1

OVERVIEW OF THE RESEARCH

1.1. Motivation

The motivation of this thesis is to design, implement and experiment a novel conceptual model that can be used to develop reliable and secure trusted remittance of service transactions without interference from third parties.

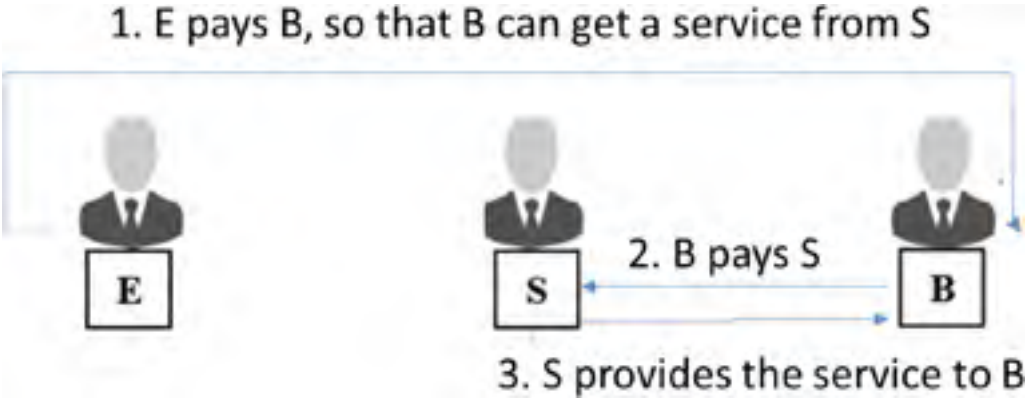


Figure 1.1: Remittance of Service (RS) provided through Remittance of Money (RM)

This research was inspired by observing remittance transactions in the city of Kinshasa. It was found that most of the transactions did not follow the initial intention of the expeditor. RM is the simple model of a remittance consisting of sending money from an expeditor (E) to a beneficiary (B). The remittance transactions observed in the field were intended to send money to a beneficiary (B) who would thereafter acquire a service from a service provider (S) as shown in Figure 1.1. In this thesis, transactions described by Figure 1.1 will be referred to as a remittance of service (RS).

Unfortunately, current money transfer operators (MTO) providing RM services are unable to offer RS transactions. MTOs stop monitoring a remittance transaction when (B) collects the

money sent by (E), and the transactions between (B) and (S) are not considered as a part of the remittance transaction.

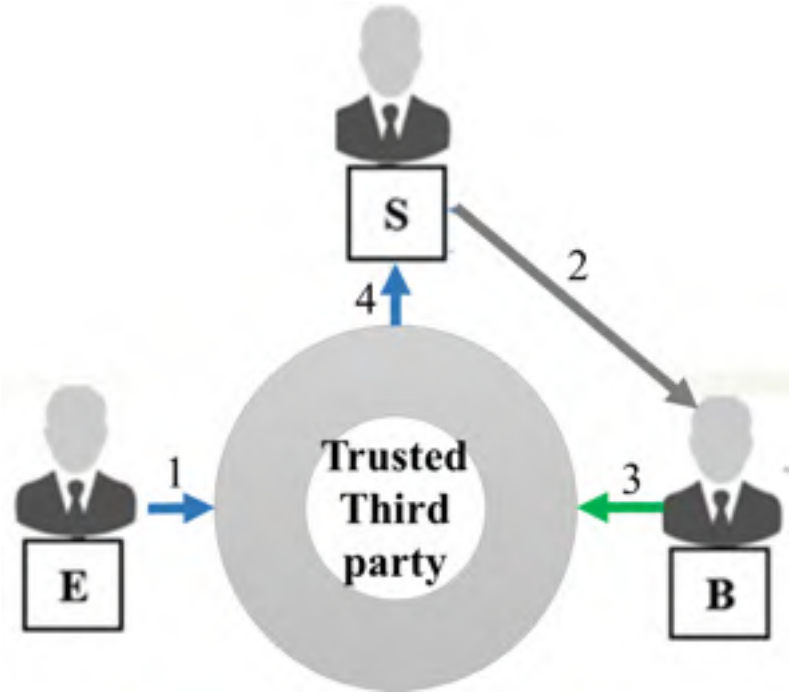


Figure 1.2: Escrow account

Legend: In blue money flow, in gray service procurement, in green confirmation.

Currently, transactions similar to RS are addressed using escrow accounts as shown in Figure 1.2. In escrow accounts, a third party other than (E), (B), or (S), and who is trusted (usually a financial institution), receives the money from (E), keeps it and then delivers it to (S) once it receives the confirmation that (S) has provided the intended service to (B).

Escrow accounts rely on a third party that is trusted by the participants so as to guarantee the effectiveness of the transaction. Nevertheless, the third party involved in the escrow account can be considered as a risk or a hindrance to the realization of the transaction itself. Usually, trusted third parties are banks or licensed financial institutions that must abide by the laws and restrictions imposed by their respective countries.

An expeditor whose family lives in a country that is under financial restrictions from the third party's country could face conditions, or sometimes sanctions, with respect to remitting money to his or her family. For example, an expeditor located in the USA cannot remit more than $1000 every 3 months to family in Cuba and commercial remittance to Syria is not allowed at all (United State Treasury, 2019). This type of government interference makes usage of escrow accounts unsuitable for the RS transactions considered in this thesis, which should not have any restrictions to transaction flows.

1.2. Problem definition

Currently, the RM model does not allow for the accommodation of RS scenarios such as the one described in Figure 1.3 below where (E) sends money to (R) with a mandate to acquire a service from a service provider (S) for himself or a third party beneficiary (B).

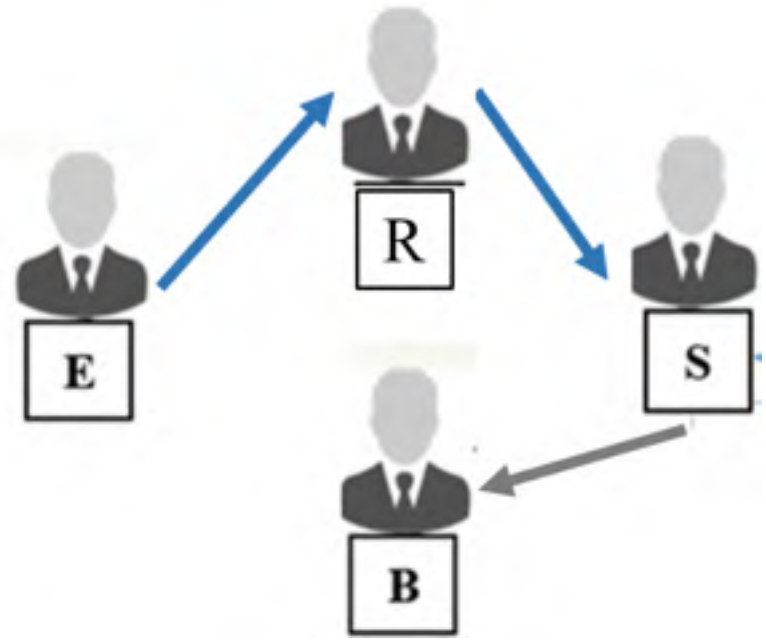


Figure 1.3: Detailed remittance of service model

Legend: In blue money flow, in gray service procurement

In the scenario above, the RM model does not discriminate between: 1) the recipient authorized to collect the money (R); 2) the service provider to be paid with the money transferred (S); and 3) the intended beneficiary of the service to be acquired with the money

remitted (B). These are 3 distinct roles with different expectations. While in certain cases these roles might be imbued to the same person, most often they are not.

The person collecting the money acts as a forwarding channel for the money to reach its intended destination, which is the service provider. The recipient of the money (R) may or may not be the beneficiary of the service provided.

For example, when a parent (E) sends money to his/her child to pay for school fees, the child is the recipient of the money (R), and in this case, also the money forwarding channel toward the school, which is the intended destination for receipt of the money (S), despite the fact that the child is the beneficiary of the service provided (B). Also, it has to be noted that even though the child is in this case the recipient and the ultimate beneficiary of the service to be acquired, he might still decide not to use the money to pay for school once he pockets it. In doing so, he will interfere with the purpose of the money transferred, as originally intended by the parent (E).

The example above shows that when an RM transaction is used to conduct an RS transaction, the entire transaction is based on two assumptions: 1) the expeditor (E) trusts the recipient (R) to pay for a service (S) for the benefit of the beneficiary (B); and 2) that expeditor (E) trusts that (B) will always honor the intended use of the money in acquiring the service provided by (S). Regrettably, these assumptions of trust made by the expeditor (E) have often proven false, with adverse consequences for both the expeditor and the beneficiary. This research intends to address this problem.

In the Trusted Remittance of Service (TRS) model that is the focus of this thesis, no assumption regarding the trustworthiness of the money recipient or beneficiary is made. A remittance transaction is considered completed only when the beneficiary receives the service paid for with the remittance money as expected by the expeditor, and, unlike in an escrow account, with the assurance that no third party will interfere. The participants transact between

themselves without third party interference using online and distributed technology with the confidence that the entire system can be trusted.

1.3. Research questions

This research originated from an informal observation of the remittance system in the city of Kinshasa where the remittance of money did not address the fundamental purpose of the remittance as intended by the expeditor, which was to acquire services (such as education, health services or the purchase of goods) for a third party beneficiary, a parent or a friend.

The research aims to establish whether this observation is confirmed or not, and, if it is the case, what remittance model could guide the development of applications that respond to this concern. The research questions are described in two phases:

1. The first phase questions:
   How prevalent is remittance in the city of Kinshasa? How easy is it to have access to it? and What do people do with the remittance they receive?

2. The second phase question:
   What could be a trustworthy remittance model that would address/include the intended use of a remittance?

1.4. Methodology

The research methodology of Abran, Laframboise, & Bourque (2003) is used to plan and present the overall approach of this research which tries to address the questions raised in the previous section. The research framework is adapted to provide empirical research in software engineering. The four phases of research are: 1) the definition; 2) the planning; 3) the development (i.e. of the original proposal and its experimentation) and 4) the interpretation of the results. Each of these four research phases is detailed in the following sections.

1.4.1. Research definition

The definition phase consists of clarifying the research to be undertaken and what is included and excluded in this work. It includes the definition of the research motivation, its objectives, a precise innovation proposal, as well as the intended users of the research findings.

Table 1.1: Research definition phase

Motivation	Objectives	Proposal	Research Users
The anecdotal report that the current remittance services do not include a protection about its intended use. Once the money is sent, it can be used for any other purpose than its intended use and the emitter has no control at that point.	• Ensure that the emitter of a remittance has control over the intended use of the money transfer. • Improve the efficiency of remittance. • Make the remittance process a more user-friendly service.	Propose a novel remittance model that includes the notion of intended use as well as allows the remittance service to be trustworthy.	Students, researchers, users of remittance services, commercial companies providing remittance services.

The next phase of the research methodology has the objective of establishing the research context and proposing the activities required and planned that are necessary before designing a novel remittance model.

1.4.2. Research planning

The planning phase identifies the research activities and the deliverables required to attempt to reach our objectives and answer the research questions. This phase began with a field survey in the city of Kinshasa. The objective of the survey was to identify the current remittance situation and better understand the problems. It is with insights from the survey results that the literature review was conducted. Table 1.2 presents the activities related to this phase of the research.

Table 1.2: Stages of the planning phase

Project Stage	Inputs	Outputs
Stage 1 Survey	• Observation of the remittance transactions in the city of Kinshasa • Survey in the city of Kinshasa	• Clarification of the research topics and unresolved issues in current remittance services
Stage 2 Literature review	Literature review on: • Remittance trends in developing countries; • The notion of trust in Distributed System concepts– its definition and goals; • Distributed Ledger Technologies definition and properties; • Ethereum Blockchain, properties, security mechanism, vulnerabilities; • DLT applications used in remittance, their business model, technologies, advantages and limitations.	• First technical report delineating the research subjects. • Literature review section of this thesis
Stage 3 Research activities	• Results of the survey and literature review • Discussion with supervisors and peers	• Proposed model to address remittance model issues • Evaluation of the model through a proof of concept • Submission of articles for publication
Stage 4 Revision and submission of the doctoral thesis	• Choice of a validation strategy	Final results • Model evaluation • Thesis submission

1.4.3. Research development

The development phase of this research describes the activities undertaken, using the literature as well as the findings from the field survey, to design the proposed 3-layered stack model. It includes the steps to model the remittance service, using a mathematical model, a trust risk analysis of the proposal as well as the definition of eight conditions to ensure the validity of the proposal. A software prototype is then conceived in order to experiment and demonstrate the potential use of the proposed 3-layered stack model for addressing the research questions.

Table 1.3 lists the activities related to this phase of the research project.

Table 1.3: Elements of the development phase

Development	Validation	Analysis
Develop the Trusted Remittance of Service model (TRS)	Publish the proposed model in a software engineering journal.	Verify comments by editors, practitioners and users interested in implementing the proposed framework in order to improve it
Develop the TRS model validation framework	Define the validation necessary to confirm the TRS model	Test the hypothesis with an experiment in the laboratory
Implement the model to a prototype and evaluate it	Prepare a case study, design and test a software prototype and execute an experiment to validate the proposal	Analyze the experiment results in order to obtain conclusions and derive improvement

1.4.4. Interpretation

The interpretation phase reviews the research problem, analyzes the solution proposed and the conclusions. It assesses the proposed solution for industry, and lastly, identifies the future work. Table 1.4 presents the elements of the interpretation phase.

Table 1.4: Elements of the interpretation phase

Results	Extrapolation	Future Work
• The research addresses the TRS model • Risks and validity conditions of the TRS model are clearly identified as part of this thesis • The validation framework of the TRS model is used to represent the performance of TRS • The experiment presented shows that the TRS is feasible to implement real world RS transactions	• Results from the experimentation based on trustworthiness and costs shows great promise	• Implement the TRS model in other languages or DLT frameworks to determine which is best suited for TRS applications • In depth business analysis of the TRS application to confirm the economic advantage of a TRS based model

1.5. Research perimeter and limitation

The research took place from January 2016 to October 2019 alternatively between Kinshasa, in the DRC to gather data, and in Montreal for discussions with supervisors for the TRS modeling and prototype validation.

Despite the fact that the survey used to derive the characteristics of the remittance business was done in the city of Kinshasa, DRC, the survey results can be generalized for any country or city receiving an in-flow of remittances.

This research focused on providing an agnostic model for developing a trusted remittance service application. While the model can be implemented using many different programming languages and within many different programming frameworks, for this research it was demonstrated, using the Kaiwen DLT decision tree technique, that implementing the model using the Ethereum blockchain provides key advantages. The choice of Ethereum and the advantages it provides are explained in Chapters 4 and 5.

There are many approaches that can be used to examine the issue of remittance, as it affects families and individuals in different aspects: commercial, societal, economic and technical, amongst others. While the model proposed in this thesis is informed by the commercial, societal and economic aspects of remittance, the primary concern is the technical software engineering aspect, i.e., how to devise a trustworthy model that could implement remittance of service applications for small transactions.

Finally, Chapter 7 explores simulation tests using the experimental prototype implementing the proposed model. It provides a limited discussion about the potential economic benefit of using the prototype. A thorough economic analysis might be needed, in future research, to decisively decide on the economic benefit of using this model, as our evaluation did not account for any potential additional overhead costs.

1.6. Research contribution

There are few examples guiding engineers who are considering the use of Distributed Ledger Technology (DLT) applications requiring a strong trust component, and that use Ethereum. This thesis provides a novel model and a process that software engineers could follow to devise Ethereum based applications related to a trustworthy remittance of service, or services, sharing similar attributes to those that provide reliable services in untrustworthy environments.

Specifically, this thesis provides the following research contributions:

1. A field survey providing a better understanding of the problem with remittance services for users in the city of Kinshasa, DRC;
2. A Trusted Remittance of Service model, a proof of concept and a rationale for choosing the DLT to be experimented;
3. A software prototype and experimentation of the proposed model using Ethereum smart contracts; and
4. An evaluation of the experimented model.

CHAPTER 2

FIELD RESEARCH

2.1. Remittance in the city of Kinshasa

At the early start of this research, a field survey was conducted in the city of Kinshasa, capital of the DRC, from September 2018 to December 2018. The objective of this field research was to understand the remittance market landscape in that country. A number of individuals using the existing remittance services were approached to better understand their user experience, the Money Transfer Organization (MTO) domain, the current environment and the conditions surrounding these remittance transactions.

This field research was done in 2 steps:

1. First, a qualitative analysis, or observation, of the remittance economy in the city of Kinshasa was conducted with the objective of identifying users of remittance transactions and the type of problems they face. It consisted of interviews and an analysis of the interview results. This first qualitative analysis informed the design of the questionnaire used in the second step of the field research, the quantitative study described below;
2. Second, a quantitative survey was done with 1007 remittance users chosen randomly from the 24 communes of the city of Kinshasa. It was executed from the 1st to the 10th of October 2018 by a team of 20 students in the computer science department of the University of Kinshasa. They used the Open Data Kit (ODK) open source survey software for the survey questions. The full questionnaire and the results of this survey are provided in Appendix 1.

Broadly, the survey questionnaire intended to answer the following questions:

1. How many types of public transport are taken when commuting to an MTO retail office to collect remittance money?
2. How much time is spent waiting inside an MTO office when collecting remittance money?
3. What is the usage of the remittance money received?
4. How frequently is remittance money received?
5. How much money is received on average?
6. Do the beneficiaries have smartphones?

The next section relates the experiences reported by beneficiaries through the results of the quantitative survey. It is followed by the reported experiences of the expeditors.

2.2. The Beneficiary experience: field survey analysis

The survey results show that receiving money in Kinshasa, from an overseas origin, in a simple and user-friendly way is, at times, a difficult activity. In Kinshasa, 59% of the beneficiaries of MTO transactions have to take at least one type of transportation, bus or taxi, to reach an MTO's retail office (Figure 2.1). Once they arrive at the remittance office, 69% report having to wait for more than 20 minutes to get served (Figure 2.2).

Figure 2.1: Number of bus or taxi to take to reach an MTO retail office (Maketa, 2018)

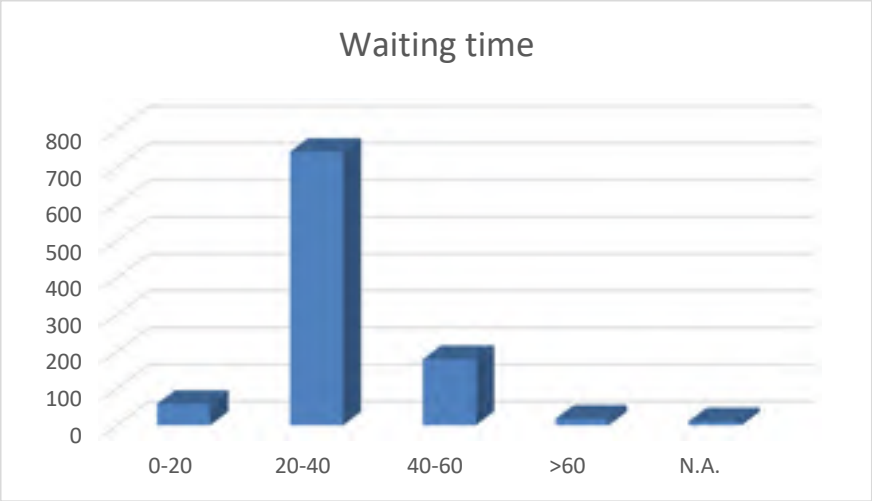


Figure 2.2: Waiting time in the MTO retail office in Minutes (Maketa, 2018)

This shows that even though the expeditor carries the money transfer cost, receiving money is not free and it is also time consuming. The beneficiary still has to disburse money and spend time to receive the money.

These overhead costs are even more significant when one has to consider that retrieving the money from the MTO is just one step in the chain of activities that the beneficiary has to perform to complete the intended purpose of the incoming money transfer. In Kinshasa, 52% of the money received serves to attend to daily subsistence needs, 26% is for education expenses, 13% for investment and 8% for health needs (Figure 2.3).

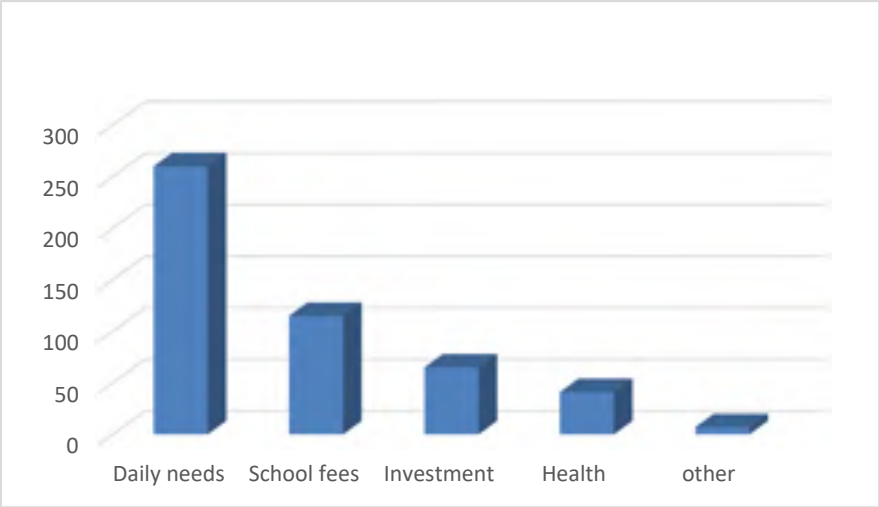


Figure 2.3: Received remittance purpose (Maketa, 2018)

That means that after retrieving the money, the beneficiary would still need to take additional transportation and spend additional time to complete the tasks for which the money is required. Considering that this process has to be repeated on a regular basis (Figure 2.4), and 42% of respondents report doing this at least once a year, we can appreciate that the beneficiary's total cost of receiving payment (TCR) may be high and time consuming.

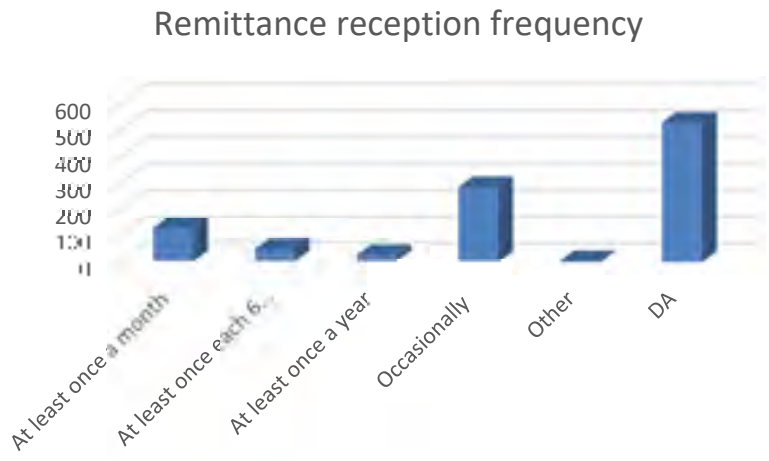


Figure 2.4: Remittance reception frequency (Maketa, 2018)

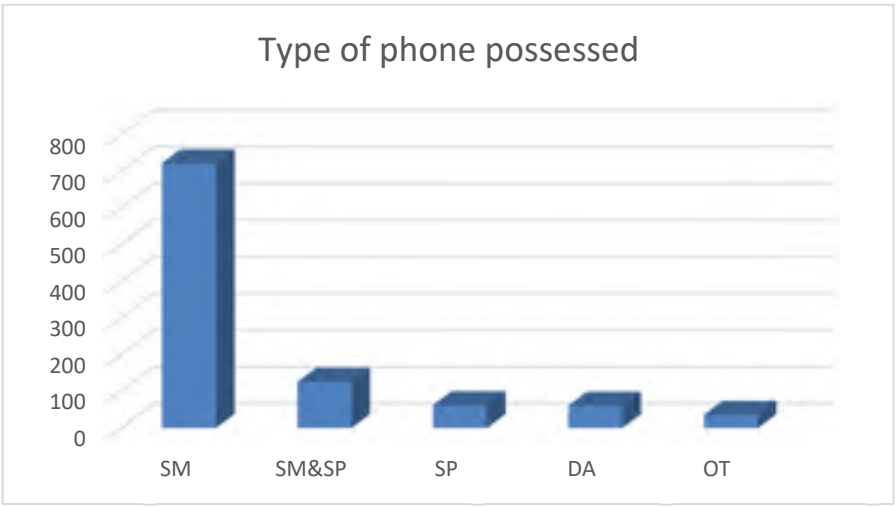


Figure 2.5: Type of phone possessed by respondent (Maketa, 2018)

TCR could be totally different if a money transfer service was offered on a mobile. As 84% of the beneficiaries of MTO transactions report possessing a smartphone (Figure 2.5), it is feasible that a mobile application will allow them to benefit from a transfer from the comfort of their own home or their work place.

It should also be noted that the average remittance transaction sent to DRC is between 100 and 500 US dollar (Figure 2.6), which is substantial in a country where a teacher earns an average

of 100 US dollar monthly. This shows the importance of remittance transactions for Congolese families.

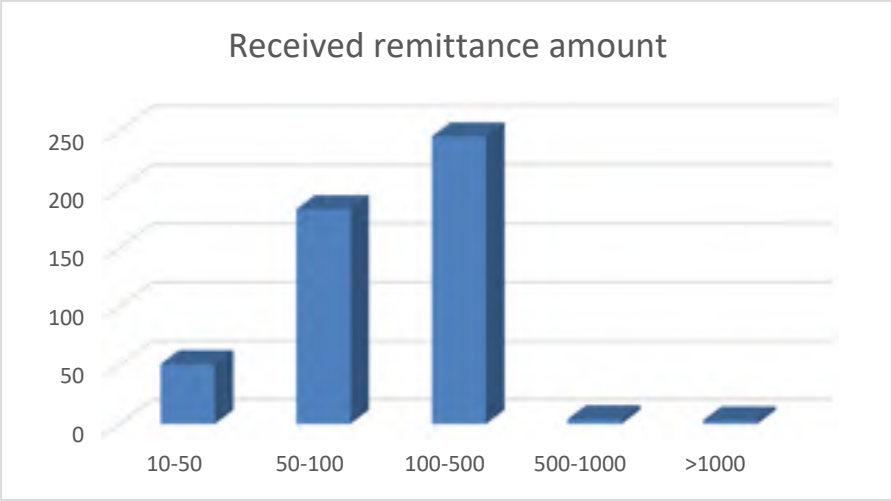


Figure 2.6: Received remittance amount (Maketa, 2018)

2.3. Remittance from the expeditor's perspective

The assessment of the remittance from the expeditor's point of view was done with interviews and desk research. This section illustrates the current situation by providing examples from transactions with Western Union, the biggest MTO market player in the DRC (Figure 2.7).

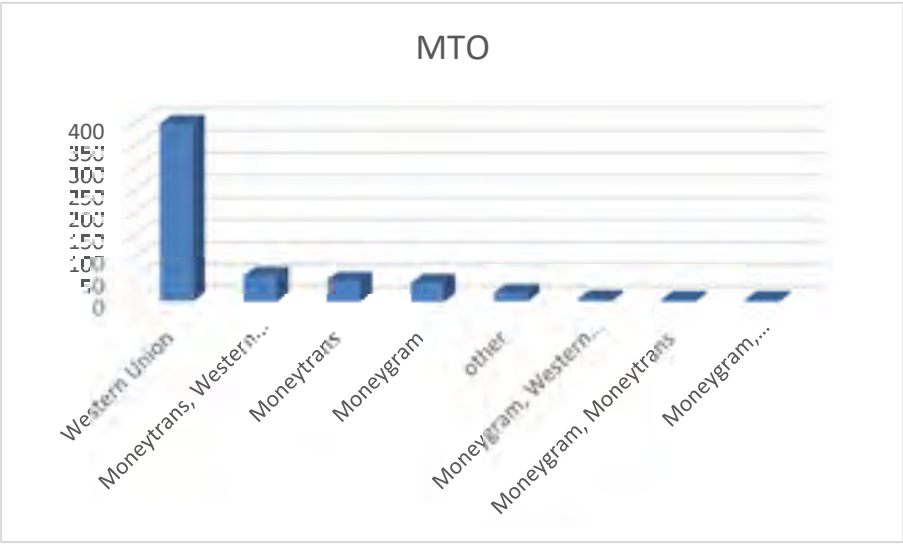


Figure 2.7: MTO market share (Maketa, 2018)

The research has shown that as there was remittance money coming from abroad to the DRC, remittance money was also being sent from the DRC to other countries. And these two groups shared the same set of concerns, although their experience interacting with an MTO was different.

The overall impression gathered from survey results is that sending money is quite easy for an expeditor from overseas sending money to the DRC. All international MTOs channeling money into the DRC provide secure websites from which expeditors can transfer money using credit or debit cards. However, the cost of an international transfer to the DRC is perceived as expensive.

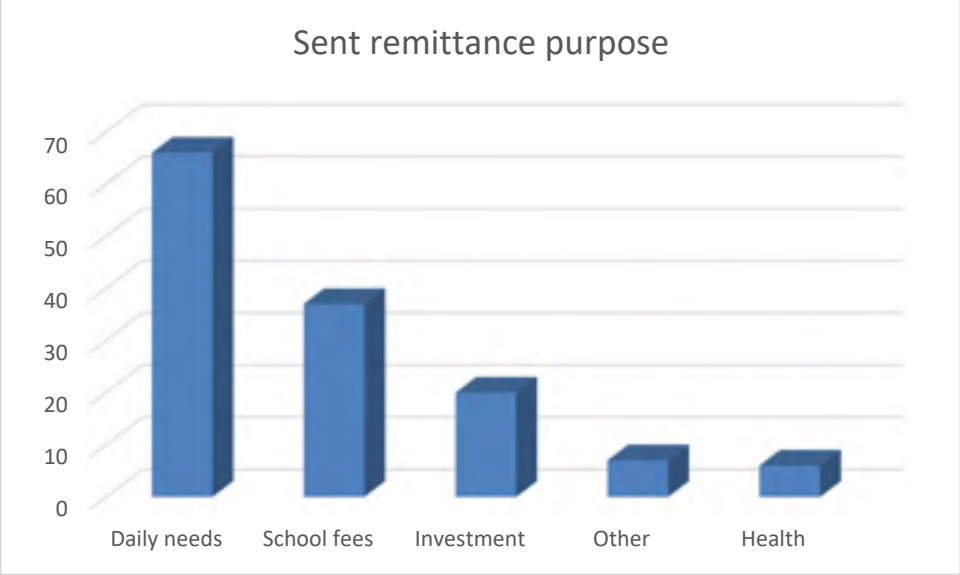


Figure 2.8: Sent remittance purpose (Maketa,2018)

Expeditors from within the DRC sending money abroad do not benefit from similar tools, mostly because the culture of using online applications and paying via credit card is not as widespread in the community and the MTOs do not see the interest in providing that service.

In Figure 2.8, the remittance purpose for expeditors from within DRC is illustrated and can be compared to Figure 2.3, which describes the purpose of remittance money received from abroad. It is interesting to see the similarity between these two figures. This implied that the cause of remittance from and to DRC are very similar and are related to providing money to family or children, to help them attend to their basic needs.

It should be noted that expeditors sending money from within the DRC share the same experience interacting with an MTO office as the beneficiary of remittance money, as described in the section 2.2. They also have to commute to an MTO retail store and wait while in there, whereas having cellphones could make their life easier.


3 Western Union also makes money from currency exchange. When choosing a money transmitter, carefully compare both transfer fees and exchange rates. Fees, foreign exchange rates and taxes may vary by brand, channel, and location based on a number of factors. Fees and rates subject to change without notice.

Figure 2.9: Western Union fine print on Exchange rate cost (Western-Union, 2019)

Expeditors from abroad sending money to the DRC, face another set of realities. In the DRC, international remittance transactions are paid for locally in US dollars because of the high volatility of the local currency, and, for transfers from countries other than the USA, MTOs incorporate currency exchange costs in addition to the visible transaction fee, as explained in the subscript at the bottom of the transfer contract (Figure 2.9). Western Union and other MTOs, receive additional benefits from transaction exchange fees that constitute another source of revenue due to the large sums of money that is exchanged every day. The exchange rate varies daily and each MTO fixes its own rate.

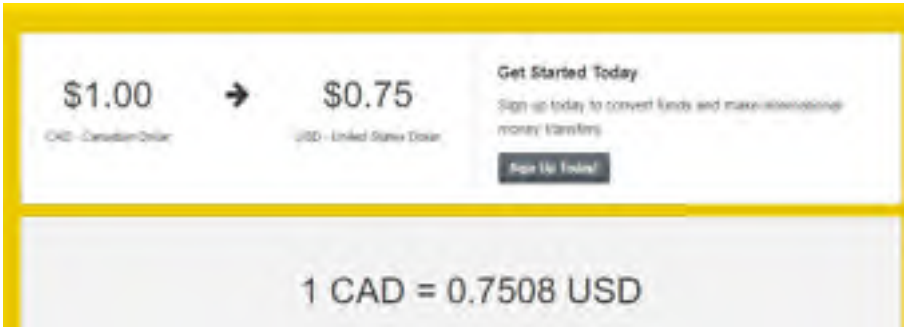


Figure 2.10: Western Union Exchange rate October 6, 2019 (Western-Union, 2019)

Figure 2.11: Xe Exchange rate October 6, 2019 (Xe, 2019)

For instance, a transfer done from Canada to the DRC, on the 6th of October 2019, shows that there is a $0.000525 USD difference between the rate displayed on the Western Union website of $1 CAD = $0.7508 USD (Figure 2.10) and the indicative rate provided by Xe, a leading exchange rate provider of $1 CAD = $0.751325 USD (Figure 2.11).

This means that for a $999.95 CAD transfer to the DRC (the maximum amount authorized to be sent online from the Western Union Canada website), the expeditor loses $0.52. That is an amount, however minimal, that should have gone to the recipient and not to Western Union.

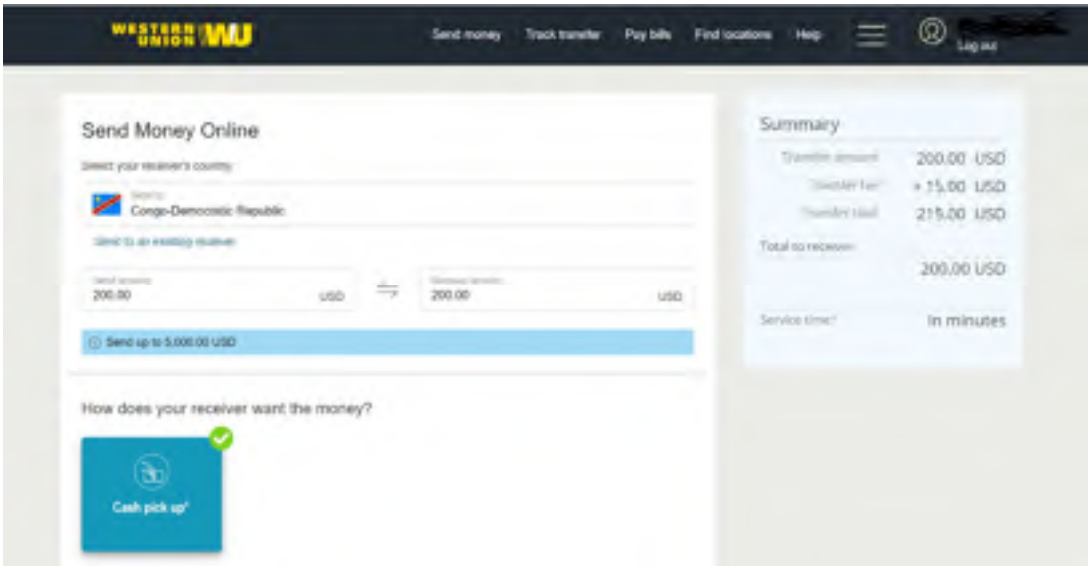


Figure 2.12: Cost of sending $200 to DRC using Western Union website on October 6, 2019 (Western-Union, 2019)

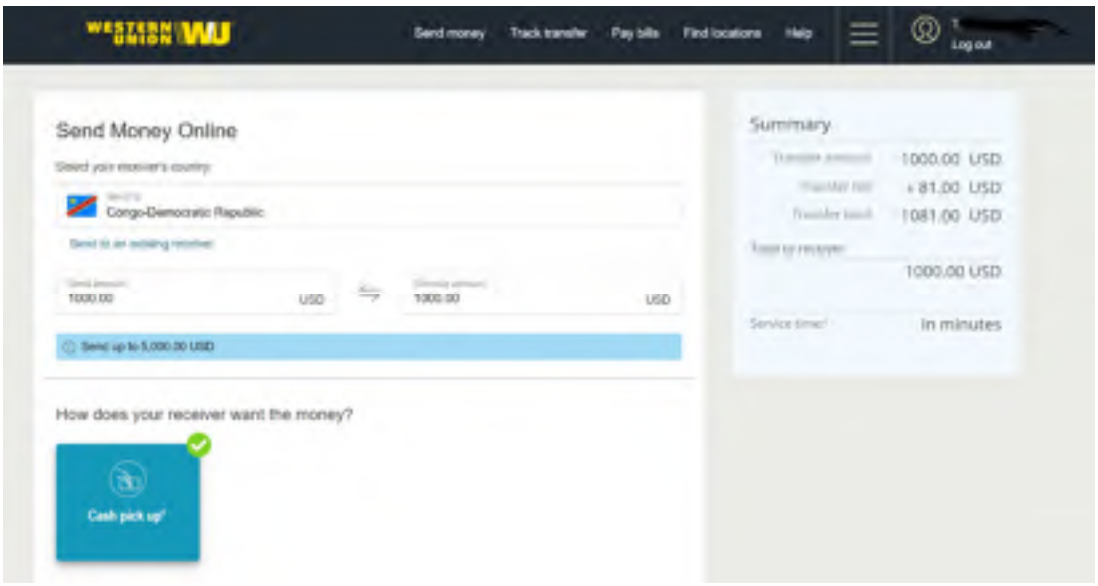


Figure 2.13: Cost of sending $1000 to DRC using Western Union website October 6, 2019 (Western-Union, 2019)

In addition to this exchange rate cost, there are transfer fees. For example, Figures 2.12 and 2.13 show that on October 6, 2019, Western Union reported charging, on its online website, a fee of $15 for a $200 USD transfer from the USA to the DRC (7.5% of the transfer amount)

and a fee of $81 for a transfer of $1000 USD (8.1% of the transfer amount) (Western-Union, 2019). That is far greater than the 3% advocated by the UN and more than the average global transfer fee of 6.1 % for 2019 (Figure 2.14).

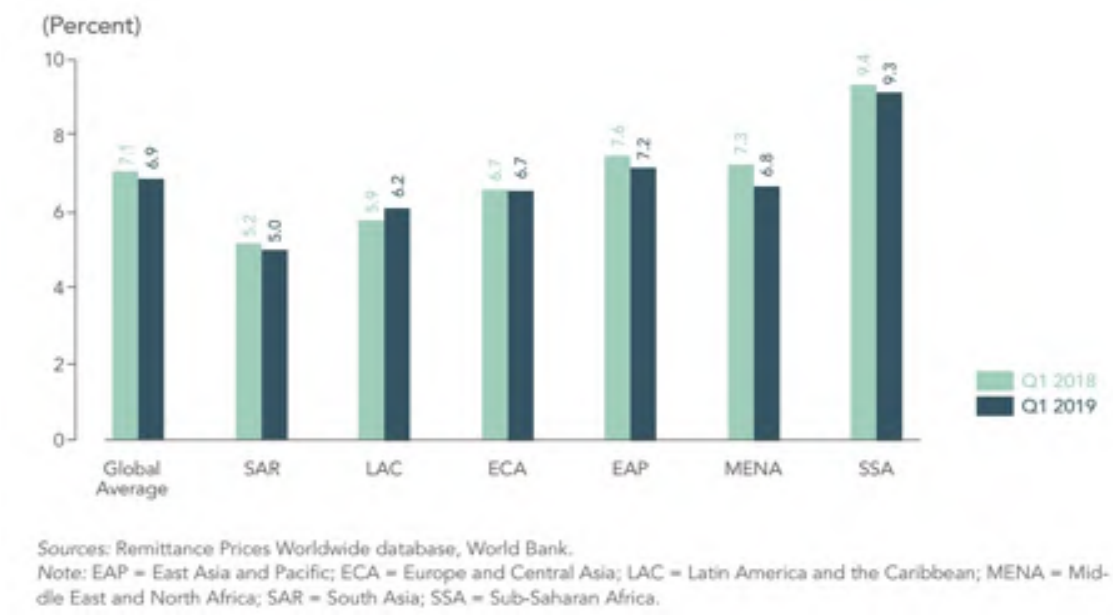


Figure 2.14: Global cost of sending $200 per economic zone (WorldBank, 2019)

Additional information was related during interviews regarding the lack of “trust” about the intended use, by the receiver, of a remittance. The expeditor described that he has no guarantee that the money sent would be used for its original intention. As soon as the beneficiary retrieves the cash from the MTO retail office, the expeditor loses control over the money and must rely solely on the good faith of the receiver to use the money for its agreed/intended purpose.

This introduces the notion of a “trust hazard” for the expeditor. The expeditor related that the recipient could perpetrate abuses that are difficult for the expeditors to discover since they live in different countries or continents.

As seen above, one can note that the average amount of money sent to the DRC is between $50 and $500 USD (Figure 2.6), which is quite substantial considering that a teacher earns only $100 US dollar monthly. 99% of that money (Figure 2.3) is sent for attending to daily subsistence needs (groceries, transport) or to pay for education and health services for the expeditors' friends and family, and the expeditor has no way to ensure that the money sent is used for its intended purpose. This applies also for expeditors from within DRC sending money abroad.

Consequently a remittance transaction is somewhat risky from an expeditor's point of view. This risk is compounded if the person who collects the money is not the intended end beneficiary; for example, a grandson collecting money for the health expenses of his grandmother.

2.4. Conclusion

In conclusion, reducing TCR for the beneficiary and reducing the trust hazard for the expeditor of a remittance transaction are the main observations of the field research.

Using the information provided by the survey, this research focuses on the issue of how to reduce the trust hazard the expeditor of a remittance perceives when he initiates a remittance of service transaction for an intended use.

This issue has been chosen because all the other concerns, such as reducing the cost of the transaction, or reducing the total cost of the remittance (TCR), can be solved by either commercial decisions, MTOs reducing their fees, or by using a mobile based application that would allow for the provision of remittance services without the beneficiary having to commute, or frequently wait in line to collect their money.

Addressing the trust hazard is challenging. There were no remittance models available at the time of writing this thesis to address this problem. Addressing the trust hazard needs a change

in the remittance model. The current remittance model as observed in the field includes as actors the money transfer operator, the emitter and the beneficiary. To address the trust hazard, the model should also include the service provider. This is a key insight that is used in the Trusted Remittance of Service model that will be seen in Chapter 5.

While attempting to solve this trust hazard, there is also an opportunity that the proof of concept that will implement the newly proposed remittance model could address the other issues raised during the field survey as well, especially issues concerning the reduction of transaction and foreign exchange costs and the issue of enhancing the user's experience regarding a remittance.

The next chapter presents the literature review of the different concepts and the current state of the art in the domain of remittances such as trust in distributed systems, consensus, blockchain, proof of work, cryptographic keys, gas and many related topics.

CHAPTER 3

LITERATURE REVIEW

This chapter uses the roadmap described in Table 3.1 to describe the different concepts discussed in this thesis.

The literature survey commences by discussing the concept of trust in distributed systems and explains that distributed systems are defined by their service delivery. Service delivery itself is dependent on the "consensus" amongst the computers partaking in the distributed system to exchange information in a trustworthy and reliable manner.

Table 3.1: Literature review road map

Concept explored	Trust in distributed systems	Consensus	DLT Ethereum Blockchain	Conceptual model design and validation	Real life DLT example
Output	Consensus	DLT Ethereum Blockchain	Proof of Work, Cryptographic keys, Gas	Lamport et al., 1982 methodology	Ripple, Stellar, Wala

Addressing consensus in an untrusted environment led to the emergence of the blockchain and Distributed Ledger Technologies (DLT). DLT constituted a response to the problem of creating trusted distributed systems without a central authority. This research limits its literature review to examine only a specific technology within the DLT family: the Ethereum blockchain. The security mechanisms that enforce the consensus synchronization at the core of Ethereum will be presented. The subsequent section reviews model correctness, as one of

the contributions of this thesis is a model. The chapter ends with an exploration of how DLTs are currently commercially used to provide remittances.

3.1. The concept of trust

Trust is an important notion in remitting value over a long distance. Trust is needed for being confident that when giving money to someone, this money will not be lost along the route toward its intended beneficiary.

3.1.1. Definition

A formal definition of trust in online systems varies amongst researchers and the problems they consider. For instance, Mui et al. (Mui, Halberstadt & Mohtashemi, 2002) consider trust to be an expectation that an agent or a system will behave without deception based on its past reputation or performance. Gandison and Sloman comprehend trust as the believed capacity of a system to accomplish its intended objective "*dependably, securely, and reliably within a specified context*" (Gandison & Sloman 2000). However, Manuel et al. (Manuel, Thamarai Selvi, & Ibrahim Abd-El Barr, 2009) contend that in distributed systems trust is an "*estimation of competence of a resource provider in completing a task based on reliability, security, capability and availability in the context of distributed environment*". On the other hand, Olmedilla et al. (Daniel Olmedilla, Omer F. Rana, Brian Matthews, & W. Nejdl, 2006) propose a definition of trust based on the service transaction: "*Trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X).*"

Figure 3.1: Triple of trust

The intuitive notion of trust is multidimensional and connected to concepts such as risk, security, reliability, and belief in the capacity to provide. There are at least two participants in an online trust relationship: a trustee – someone, an application or an entity – who is trusted to provide a service, and a trustor – someone, an application or an entity – who gives trust in expectation of a service. Between the two there is a transaction or a trust relationship (Figure 3.1).

When the online transaction involves a person as trustor, the trust embedded in the triple relationship encompasses an additional degree of unpredictability, anxiety and vulnerability for the trustor. Communication delays, absence of acknowledgement, and lack of familiarity between parties can be perceived as an "incomplete or distorted disclosure" of information by parties to the transaction and as part of a "calculated effort to mislead, distort, disguise, obfuscate, or otherwise confuse" (Li, Pieńkowski, van Moorsel, & Smith, 2012). The remittance system proposed with this research intends to remove this vulnerability for the trustor.

3.1.2. Trust in distributed systems

Distributed systems, as the one this thesis considers, offer their services across computers exchanging information and consuming resources throughout the entire world. For example, crypto money networks offer payment services, e-commerce web sites provide sales service and the IOT network of sensors provides data collection service (Manuel et al., 2009; Daniel Olmedilla, Omer F Rana, Brian Matthews, & Wolfgang Nejdl, 2006)

As such, the notion of service is intrinsically linked to a distributed system; it is its purpose. Trust in the system to deliver a service is what the user of the system expects.

Maintaining reliable service delivery in a distributed system requires a flawless combination of two major dimensions of trust security and service delivery (Daniel Olmedilla et al., 2006). The security dimension of trust deals with "classical" security challenges for which a large body of research exists. It centers on ensuring that the right people have access to the right

resources, and that the overall system is preserved from theft of confidential information and malicious attack.

Service delivery focuses on the ability of the trustor to transact with the trusted system without the fear of being deceived either by the trusted system, a malicious attack or by a system failure. It encompasses notions such as reliability, capability, availability, dependability, the reputation of the participants to the system, and consensus on what the system has to consider as the shared set of valid transactions.

For this research, dimensions such as reliability, capability, availability and dependability of the distributed system depend on the system code quality and are managed by the theories of software engineering and quality assurance. Aspects such as the reputation of the participants are not relevant to this research since part of the requirements for the distributed system we consider is that it must accept any willing participant, irrespective of a good or bad reputation. Consequently, the dimension of trust related to the distributed system of this research is the dimension of consensus, or how nodes of a system spanning globally share a unique state and protects itself from malicious hacking.

After consensus is considered, it is important to ensure that it is used in an effective manner without creating additional breach of trust. For that, the use of an abstract conceptual model that mathematically or logically proves the model correctness and reliability is important. The modeling activity has as objective to demonstrate that the proposed model is impervious to faults and can handle the level of expected transactions. This is the approach this research will pursue and that will further be explained in section 3.3.

3.1.3. Conclusion

In conclusion, the trusted remittance of service as defined for this research will be delivered through a distributed system without a central authority. Therefore, the trusted system must reassure the trusting participants that it can reliably provide its service while accepting any

participant, even those with clearly nefarious motivations. DLT systems, especially blockchains, have been designed to address these types of requirements.

While originally built to transact money on untrusted environments, the technology is increasingly used to tackle more complex scenarios. For this research, the Ethereum blockchain has been used. The next section delves into the details of how DLT, blockchain and Ethereum manage consensus amongst their distributed nodes and untrusted participants.

3.2. DLTs

Building a secure and globally scalable RS system and implementing it is not trivial. Fortunately, in the last decade, distributed ledger technology (DLT) and its most recent flagship technology called blockchain have emerged and introduced new platforms for remittance systems based on cryptographic keys and complex synchronization mechanism called consensus.

The term distributed ledger technology comes from the accounting analogy; a company's accounting ledger contains all the ordered financial transactions that need to be accurate and secure. Blockchains are one type within the larger DLT family. They are based on a structure that is a chain of data blocks, thus its name. Other data structures or technologies can be used to deploy digital ledgers, amongst them: the directed acyclic graph used in the tangle protocol of IOTA (Popov, 2017), hash graph tree used in Swirlds (Baird, 2016) or the byzantine fault tolerant algorithm used in Ripple (Ripple, 2017) and Stellar (Mazieres, 2015). The uniqueness of DLT technologies is that they can perform traditional remittance services without the need to involve a third party organization, such as a central bank authority, to guarantee the money transfer.

DLT can be private when deployed by a private organization for its internal use, or it can be public when it allows a global audience to participate. Public DLT networks are not necessarily trustworthy environments, i.e. no one can deny another access to the network. As

a result, a participant's motivation and intention are unknown. Therefore, by default, participants in the system should not be trusted given the real possibility of malicious participants trying to cheat the system and defraud legitimate users access to their money. In this unsafe environment, it is the DLT protocol that guarantees the validity of the participants' transactions using cryptography and complex consensus mechanisms.

In addition to guaranteeing trust, the latest DLTs permit the development of software programs allowing implementation of complex transaction scenarios and decision logic. These complex scenarios and decision logic allow to move beyond just remitting money and start to remit value for anything that can be modeled as a transaction between two or more participants. Software running on DLTs are called Smart Contracts. Once deployed on the DLT, a smart contract cannot be changed. The term smart contract originates from the understanding that it is immutable, similar to a legal contract. As a result, once deployed, the smart contract must be enforced, as would a legal contract that has been signed.

Choosing the right DLT for a particular application is essential as each DLT comes with its own set of features and limitations. This thesis chose its DLT using the Kaiwen DLT tree (Kaiwen Zhang, 2019). It is a simple questionnaire (5 questions) that will be used in Chapter 5.

3.2.1. Ethereum blockchain

As stated above, the word blockchain is a generic name applied to all the DLTs using the chain of block data structure. There are many different blockchain-based DLTs. The Bitcoin cryptocurrency's blockchain was the first blockchain to enter production in 2008. From its success, many alternative blockchains emerged, each one addressing the perceived limitation of the original Bitcoin blockchain: Litecoin, Montero, Ethereum, and many others colloquially called "alt-coin".

3.2.1.1. Blockchain description

A blockchain is “a network-centric protocol and platform for recording and transferring ownership and trust on a peer-to-peer basis” (Antonopoulos, 2017). It takes its name from its underlying data structure composed of blocks containing data and transactions that are linked between them by cryptographic hash references (Figure 3.2).

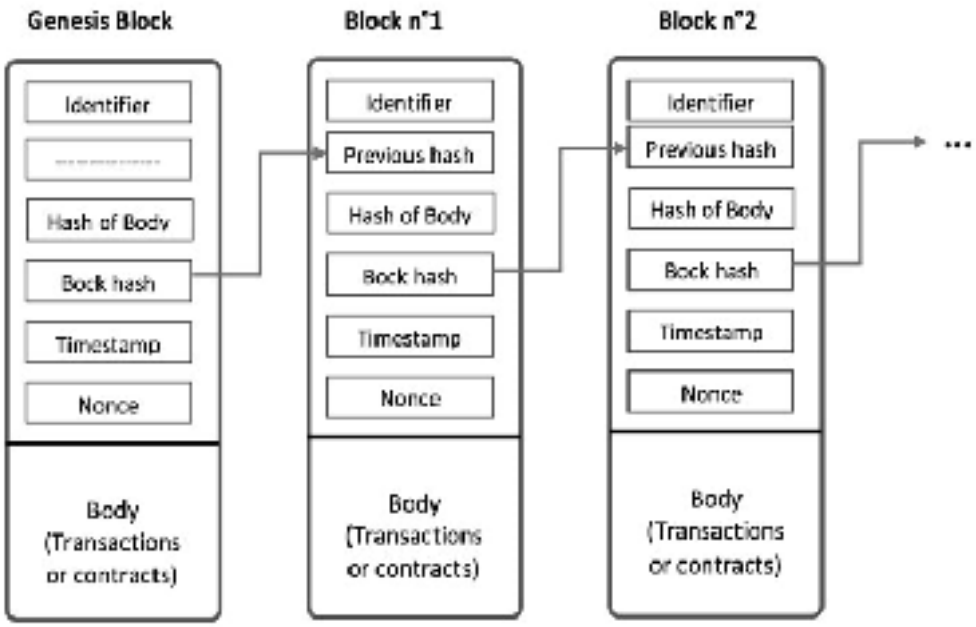


Figure 3.2: Block structures (Wood, 2019)

The blockchain behaves as a completely traceable digital chain of ownership (Peng, 2015), and it can be considered as a transaction-based state machine. It begins with an origin (genesis) state and incrementally accepts transactions to mutate into a current state. The current state is the one accepted as the canonical “version” of the world of Ethereum (Wood, 2019). The state of the blockchain at any given time is stored into blocks.

Formally:

$$\sigma_{t+1} \equiv Y(\sigma_t, T) \qquad \text{(Wood, 2019)} \qquad (3.1)$$

where σ_t is the state of the blockchain at time t, T is the list of transactions and Y is the blockchain state transition function. Y allows components to carry out arbitrary computation, while σ allows components to store the arbitrary state between transactions (Wood, 2019).

However, while ownership is traceable, in public blockchains the real identity of the owner is not. Cryptographic keys are used for uniquely identifying participants and often participants obfuscate their real identity behind their publicly visible cryptographic keys.

A blockchain can be considered as a distributed database that only allows reading and writing operations; update and delete operations are not possible at all (i.e. read and write operations are permanent). Blockchains possess 3 key features: decentralization, consistency and scalability (DCS). Decentralization ensures blockchains remain reliable without requiring a trusted third party. Consistency ensures that data are always identical throughout all the nodes of the system, and scalability guarantees that performance and availability of the system is well maintained when there is an increasing number of nodes joining the system (Kaiwen Zhang & Jacobsen, 2018)

3.2.1.2. Consensus and performance

The Bitcoin blockchain reestablished the "consensus" as one popular model for data synchronization in decentralized peer-to-peer nodes, and this despite its widely perceived poor scaling performance. Before the Bitcoin blockchain, distributed consensus was perceived as "*a synchronization primitive to be used only in applications in desperate need of consistency and only among few nodes*" (Cachin, Kursawe, & Shoup, 2000; Vukolić, 2015). Today, Bitcoin and its underlying blockchain benefit from wide acceptance, and increasingly impose themselves as "the cryptocurrency". Its main constraint is its ability to keep performance

satisfactory while accepting an increasing number of nodes and transactions. This constraint extends to all consensus-based blockchain systems, including Ethereum.

Consensus strategy is the main limitation to consider regarding the performance of a blockchain in terms of latency and the amount of processed transactions by unit of time (throughput). A consensus-based system performs well when the number of nodes is small; this changes when the number of nodes increases. Blockchains using a consensus synchronization strategy scale up globally but with a deterioration of performance.

Bitcoin's theoretical network peak transaction speed is 7 transactions per second (TPS) (Gerard, 2017). Ethereum's theoretical maximum TPS is 22. In July 2019, the Ethereum network did not exceed 11 TPS, dropping on several occasions to around 7 TPS (Magas, 2019). These performances have to be compared to credit-card payment systems that approximately serve on average 2000 transactions per second, (Zohar, 2015).

3.2.1.3. Type

While most of the blockchains provide cryptocurrency or some sort of payment functionalities, certain types of blockchain do not. Hyperledger, for instance, the Apache foundation open source blockchain backed by corporations such as IBM, does not provide money transferring functionalities and targets business-oriented applications of blockchain technologies.

New blockchain infrastructure, colloquially called blockchain 2.0, attempts to address the performance issue by replacing the slow Bitcoin Proof of Work consensus algorithm with improved approaches such as Proof of Stake, Proof of Elapsed Time, Federated Byzantine Agreement (and others). Blockchain 2.0 also provides computational capabilities, such as Ethereum, that emerged from the research to add computational capabilities to the Bitcoin blockchain (Vitalik Buterin, 2013).

Off-chain payment channels is another strategy considered to speed up TPS. This strategy is used by the Lightening Network (LightningNetwork, 2017) for Bitcoin or Raiden (Raiden, 2019) for Ethereum.

The blockchain technologies (Bitcoin, Ethereum, alt-coin) emanated from a long process that started at the beginning of the internet era in the 1980s when the self-named, “cypherpunks” group sought to use cryptography technologies developed in the 1980s (e.g., public key encryption and PGP) to ensure total privacy when interacting and shopping online. Cypherpunks wanted to build a universal currency for payment, not relying on states or a central authority, which could ensure total privacy. Cypherpunks feared that their payment transactions recorded in a financial institution’s server logs could be sold to data brokers, or expose them to unwanted scrutiny (Peck, 2012).

3.2.1.4. Origin: the double spending problem

The major issue preventing the creation of a universal payment system was the double spending problem. The double spending problem consists of the difficulty to guarantee that in a totally decentralized and anonymous system, without a single supervising authority, every amount of currency possessed by anyone could not be spent more than once. In centralized systems, like credit card systems, it is the central authority that guarantees no double spending occurrence.

The blockchain solved the double spending problem by using cryptography and peer-to-peer consensus mechanisms (e.g., Proof of Work, Proof of Stake, Proof of Elapsed Time, Byzantine fault tolerant consensus).

3.2.1.5. Double spending problem as an example of the Byzantine Fault Tolerance problem

The double spending problem can be seen as a particular case of the most general Byzantine Fault Tolerance problem (BFT). In a BFT problem, the objective is to synchronize message (transactions) exchange between nodes while acknowledging that some nodes might be sending deceptive messages, either willingly with malice or unwillingly due to temporary or permanent malfunction.

In such a system, a node cannot know whether it has received all the messages exchanged in the system and it cannot a priori ascertain whether a message it receives is an honest or malicious one, even when the message is presented according to the required protocol.

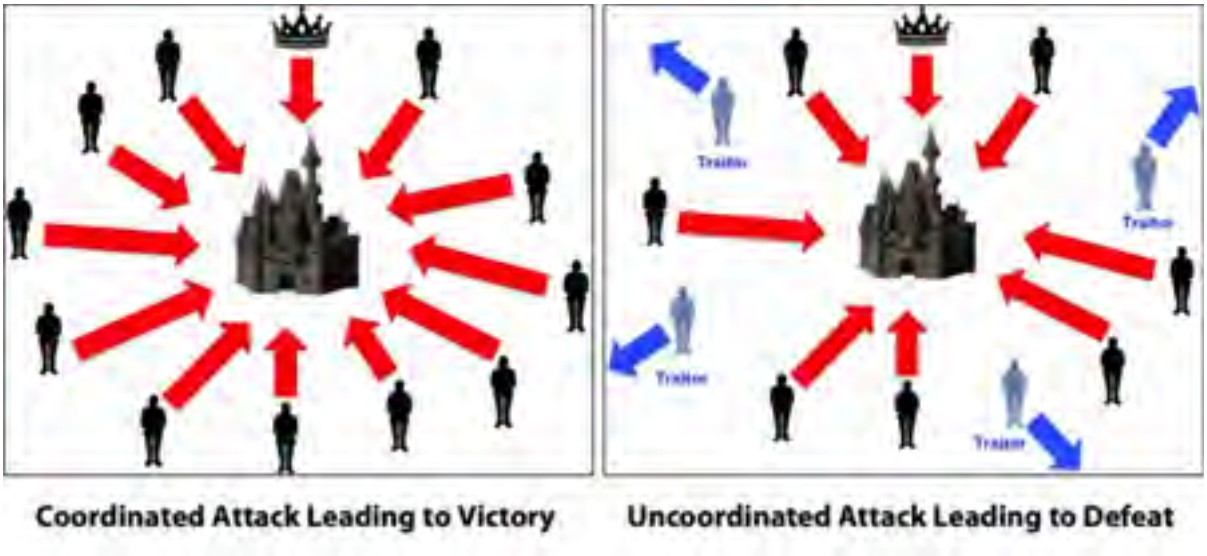


Figure 3.3: Byzantine General (BitcoinWiki, 208)

The name BFT came from the thought experiment of the Byzantine generals coordinating an attack against a city around which they were posted. All roads linking the generals between them pass through the city and these generals have to reach a consensus on when they will attack (Figure 3.3). Every messenger they send between themselves to coordinate

the time of the attack passes by the targeted city and incurs the risk of getting caught and the message being destroyed or maliciously altered. To add more complications, some of the generals are traitors and will send misleading messages to their peers to avoid them reaching consensus on the timing of the attack.

It has been proven by Lamport et al. (Lamport, Shostak, & Pease, 1982) that for a BFT problem with N malicious nodes, a consensus can be reached if there are 3N +1 honest nodes. But, if signatures are used to authenticate the exchanged messages, a consensus can be reached with 3N honest nodes. This is a result that will be used for the design of a 3-layered stack TRS model proposed in Chapter 4.

3.2.1.6. How did the blockchain solve the double spending problem?

The double-spending problem can be re-stated as the problem of ensuring that all participants in a network agree on a single truth, in this case represented by a transaction log history. Sharing the same history of transactions prevents different versions of the transaction log to be shown to different participants and therefore prevents certain transactions from being repeated by certain participants. This averts double-spending as well as averts history from being edited (Chain.com, 2014).

The concepts that led to the resolution of the double-spending problem in a peer-to-peer network, and that subsequently led to the creation of the blockchain, began to appear in 1998 with the publication of Wei Dai's theoretical "B money" model. Dai proposed a currency whose value depended on a yet undefined "computing effort", meaning that *"The number of monetary units created is equal to the cost of the computing effort"* (Dai, 1992). Dwork and Naor (Dwork & Naor, 1992) were the first to connect the idea of computing effort to the cryptography that they applied to deter spamming. They proposed usage of a cryptographic proof of computational expenditure as a means of transmitting value over the Internet instead of using currency (Wood, 2019).

Nick Szabo (Szabo, 1997) proposed his own theoretical model named the “bit gold”. In this model, Szabo proposed the concept of a “proof of work” or “puzzle function” that further detailed the concept of “computing effort” or “computational expenditure” proposed by Dwork and Naor (Dwork & Naor, 1992).

A “puzzle function” is a mathematical function that is costly to compute but whose solution is easy to verify. The computation of the puzzle function requires investment in costly specialized computer hardware, making it expensive for an attacker to compromise the system. Szabo suggested that a special node with boosted processing power would compute the solution of a puzzle function and attach the solution to a transaction to confirm it has been validated.

Subsequently every member of the network would have to check the function’s solution (e.g., the proof of work) before accepting a transaction. Back (Back, 2002) later produced a similar system.

Vishnummurthy et al. (Vishnumurthy, Chandrakumar, & Sirer, 2003) were the first to propose a system protected by a proof of work mechanism augmented with digital signatures and a ledger in order to ensure that the historical record couldn’t be corrupted and that malicious actors could not spoof payment for a peer-to-peer file exchange system (Wood, 2019).

3.2.1.7. Satochi Nakamoto and the creation of the blockchain

In 2008, Satoshi Nakamoto (Nakamoto, 2008) extended the proof of work concept to the Bitcoin protocol. The Bitcoin protocol is built upon the concept of using computing power and a digital ledger to protect transaction integrity, and prevent double spending, as we have seen in the previous models. However, it added a key innovation; a public, peer-to-peer decentralized database based on a data block structure, which would support Bitcoin transactions, the first blockchain proposal.

Nakamoto's proof of work introduced the concept of mining (Figure 3.4). Mining is based on the principle that amongst all participating nodes to the decentralized blockchain network, there is a special category of nodes called "mining nodes" with the responsibility of solving the puzzle function, bundling transactions into blocks and validating blocks. Of all the mining nodes, there will always be, at any given time, only one mining node, the leader, that decides on the next valid block to add in the blockchain and broadcasts it to the entire network. The leader mining node is not known in advance, it is chosen by a competition, and the leader mining node is the one that solves the puzzle function the fastest.

The solution to the puzzle function is a scalar value, called "nonce" that the mining node attaches to the block it has validated. Every node in the network will use the nonce to verify whether it solves the proof of work. Etash is the Ethereum proof of work algorithm used to compute the nonce by the mining node, and used to verify the nonce by participating nodes (Vitalik Buterin, 2013; Wood, 2019).

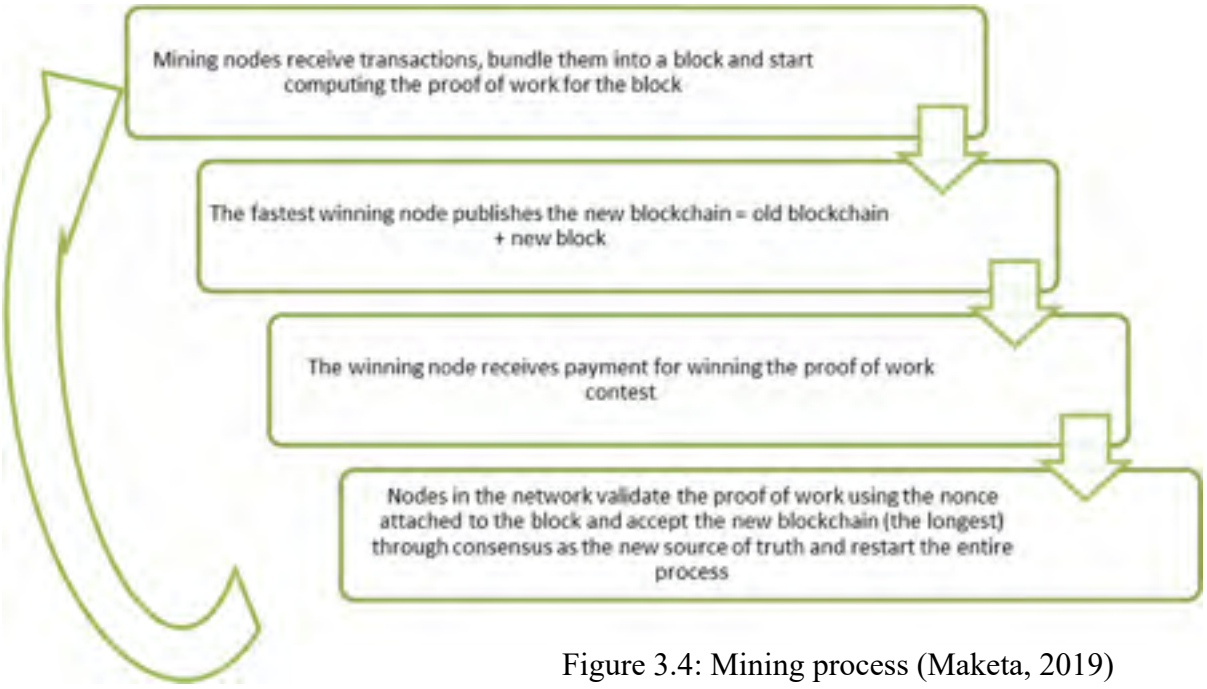


Figure 3.4: Mining process (Maketa, 2019)

Proof of work ensures: a) message ownership, b) message integrity c) message transaction history, and d) message trustworthiness by discouraging malicious nodes to send deceptive messages by making it costly for an attacker to pervert the system.

Ethereum blockchain proof of work computation is probabilistic and not deterministic; there is no way to know in advance which nonce will provide the solution, all numbers have to be tried successively until one is found by brute force. Rapid hashing nodes might try more combinations in a shorter period of time and they might find the solution sooner, or they might not. However, for the same hashing capacity, two nodes have the same probability of finding a nonce and they do not know in advance how much time it will take. The process is similar to a lottery.

The difficulty of the puzzle algorithm is self-adjusting so that on average, a new block is created every ten (10) minutes across the entire worldwide Bitcoin network. Every ten minutes, a winner is announced (i.e. a new block is created) and the competition resets. All the nodes get back to try to win the new round of the competition and create the next block.

3.2.1.8. Blockchain data integrity

To compromise a blockchain network and rewrite its history of transactions, an attacker would have to: 1) re-compute the hashes for the historical transaction it wants to change, beginning from the block it wants to change to the last current block of the blockchain, and 2) incrementally re-compute the proof of work for all these blocks in succession.

This would need the deployment of an enormous amount of processing power in a very short period of time, because while the attacker is re-computing hashes and proof of work, additional blocks are appended to the blockchain.

Still, theoretically any group of people controlling 51% of the computing power across the network would be capable of doing that. That is the reason the mining race has been designed as probabilistic, so as to ensure that no one entity could take control of the blockchain by winning the race most of the time and be able to rewrite transaction history (Chan, 2016). In

the Bitcoin system, the longest blockchain is the unique copy of the truth; all the unspent transactions output logged on that blockchain are accurate.

3.2.1.9. Cryptographic keys, blockchain address and digital wallet

In the blockchain, nodes are uniquely identified using an address. The blockchain address is the public key part of an asymmetric private/public key pair. An example of a public/private key pair is provided in Figure 3.5. This pair is generated by MyEtherWallet.com and the address corresponds to the public key, the private key is provided just below.

Figure 3.5: Example of public/private key generated in MyEtherWallet.com

Asymmetric public and private keys are linked by a function F with the following properties:

$$F(\text{Private key}) = \text{Public key} \tag{3.2}$$

$$\text{Private key}(\text{message}) = \text{signature or encrypted message} \tag{3.3}$$

$$\text{Public key}(\text{signature}) = \text{message} \tag{3.4}$$

The F function ensures that only one private key can generate one public key. There is no known mathematical way to compute a private key from a known public key, where these keys having the properties described in the relationships 3.2, 3.3 and 3.4.

Asymmetrical keys can be used to encrypt data or sign data depending on how their properties are used. In the blockchain protocol, we used the property to sign the data using relation 3.4 in the manner described in Figure 3.6. The private key linked to a blockchain account maintains non-repudiation by signing every transaction and every message it transmits over the network. Blockchain digital wallets only hold cryptographic keys, both the private and the public keys. The digital wallet does not hold money. With Ethereum, money is stored in the blockchain account. Without the private key, it is impossible to recover access to saved money linked to a corresponding public key blockchain account. This mechanism ensures that no one can gain access to a private key using a blockchain address (public key).

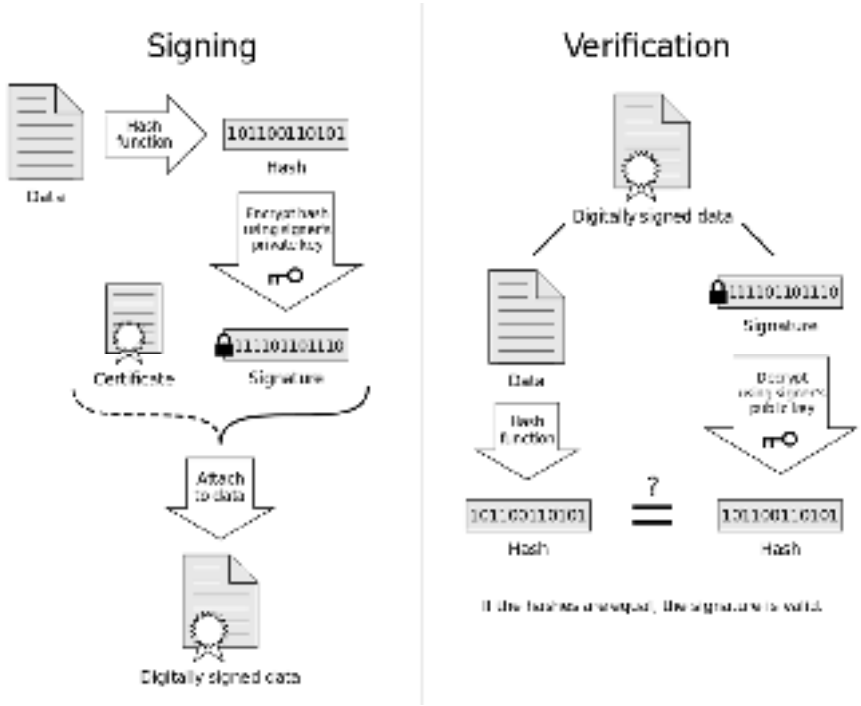


Figure 3.6: Private and public key signature in blockchain (Acdx, 2008)

The SHA-256 hash algorithm is also used in the proof of work algorithm.

3.2.1.10. Ethereum Virtual Machine and gas

The main difference between Ethereum and the Bitcoin blockchain is that Ethereum has a Turing complete computational engine, the Ethereum Virtual Machine (EVM), which allows for the development of a software application called a smart contract. Buterin (Vitalik Buterin, 2014) and Wood (Wood, 2019) proposed the technical specifications that evolved into the current Turing-complete EVM specifications.

Ether is the cryptocurrency of the Ethereum network in the same way that Bitcoin is the cryptocurrency of the Bitcoin blockchain. Ether comes in different denominations. The smallest basic unit of the Ether is called Wei in honor of Wei Dai, the author of the precursor of the proof of work concept: 1 Ether = $1x10^{18}$ Wei.

Every node in the Ethereum network runs one EVM. Collectively, all EVMs can be considered as one global computer running on all the nodes of the network at the same time. Smart contracts run inside the EVM with no restriction on the type of operations they can perform as long as there is Ether in the smart contract account balance.

Smart contracts cannot be modified; they always execute the same code and for a public network, this code is also public. Anyone can see the code of the smart contract and analyze it for bugs to exploit. Therefore, special care has to be provided when coding smart contracts, as any coding error is final and cannot be corrected. Smart contracts control their own Ether balance and their own key/value store of persistent variables (Vitalik Buterin, 2013; Wood, 2019).

Ethereum has a close relationship to its cryptocurrency. In the Ethereum network, nothing is free; everything that uses EVM resources has to pay a gas cost in Ether. Every Ethereum transaction, smart contract creation or smart contract function call consumes gas. Smart contracts and developers have to pay gas for the calculations in the EVM, and users have to pay gas to access and interact with smart contracts.

The amount of gas consumed for processing a function or a smart contract in the EVM depends on the ‘opcode’ contained in the function or the smart contract. An ‘opcode’ is a low-level function running in the EVM, such as data storage, read, write, addition or multiplication (Wood, 2019). Each ‘opcode’ has a gas value associated to it (see Table 3.2).

The Ethereum yellow paper defines the cost of gas related to each ‘opcode’. Changes to these relationships are managed by Ethereum Improvement Proposals (EIPs). For example, the EIP 150 added 400 gas to functions used for denial-of-service attacks, making these functions more onerous for an attacker to use (Vitalik Buterin, 2016).

Table 3.2: Example of Gas Costs, Ethereum Yellow Paper (Ryan, 2018)

Value	Mnemonic	Gas Used	Notes
0x00	STOP	0	Halts execution.
0x01	ADD	3	Addition operation.
0x02	MUL	5	Multiplication operation.
0x03	SUB	3	Subtraction operation.
0x04	DIV	5	Integer division operation.

The expeditor of a transaction has to state upfront what the amount of gas (gas x gas price) he is willing to pay for the transaction to be executed in the EVM.

If a transaction is completely executed using less gas than the specified amount, the transaction runs normally and at the end of the execution, the transaction’s expeditor receives a gas refund and the miner of the block receives the gas fees. If the quantity of gas proposed is insufficient and the transaction runs out of gas, the entire execution is canceled. However, the gas consumed until interruption is transferred to the miner and is lost to the transaction’s expeditor. It is the expeditor’s obligation to ensure he has the right amount of gas to cover the EVM execution.

The intent of the gas system is to discourage malicious nodes, or hackers, to overload the Ethereum network with malicious transactions without incurring a cost. An attacker will have to pay proportionately for every resource that the attack will require. The gas system also discourages programmers to code overly complex or poorly written smart contracts that drain EVM resources.

The notion of gas is one of the most important concepts to understand and it is the major constraint to account for when developing efficient Ethereum applications. There is an entire gas economy in the Ethereum blockchain, with miners adapting to the minimum amount of gas they are willing to accept based on market congestion. During high congestion periods or when there are problems in the network, the gas cost requested by miners augments.

This means that calculating the minimum amount of gas required based only on the smart contract 'opcode' table is, sometimes, not sufficient. One also has to consider the "gas inflation" due to network conditions (Schiener, 2015). Websites such as eth gas station (https://ethgasstation.info/index.php) provide information on the gas prices on the blockchain and allow for proper calibration of the gas price for every transaction.

While an effort has been made to develop gas efficient code for the TRS application, this thesis did not concentrate on gas code optimization. However, this notion of gas has an implication for the user of the TRS application, as will be explained in Chapter 7.

3.3. Correctness of the service delivery model

Public DLTs have the particularity that the entire model, algorithms and source code is entirely exposed to the public. Hackers can review the logical model of the entire application and the source code of the smart contract, and analyze them for logical and technical vulnerabilities to exploit.

Smart contracts with logical and technical implementation flaws have caused millions of dollars in theft and caused the first major technical update of the Ethereum network. That led smart contract bug analysis to be raised to an integral part of DLT software development.

Defects can be present in the logical model or within the language implementation. In DLT, most of the correctness effort has focused on the vulnerabilities associated with the software development language. For this thesis, the language used for experimentation is Solidity, the most commonly used Ethereum development language. Dingman et al. reviewed Solidity technical vulnerabilities (Dingman et al., 2019). A list of some of the most well-known Solidity vulnerabilities is available and the community maintains several pages of publicly available information about Solidity coding exemplary practices (Verification, 2019).

To address vulnerability related to Solidity, this thesis compiled some of the most known Solidity vulnerabilities exposed in the sources above and included them in its code test suite using Selenium (Selenium, 2019), a web automation framework. Selenium enabled us to simulate interaction with a web application and 80 scenarios, ranging from code bug detection to known solidity vulnerabilities, were tested on the application code.

This thesis however, proposes the Trusted Remittance of Service model (a conceptual logical model) that can be implemented in any ledger technology, not only with Ethereum. It is therefore important to ensure that the conceptual logical model itself is impervious to logical flaws. To demonstrate the logical correctness of a model, we used the types of correctness demonstration established by Lamport et al, Dai, and Szabo (Dai, 1992; Lamport et al., 1982; Szabo, 1997), who, framed the theoretical concepts necessary to develop DLT well before the technology itself existed.

In particular, the approach used to prove the correctness of this thesis model is inspired by the method used by Lamport (Lamport et al., 1982) to find the solution to the Byzantine general problem. This thesis also uses one of the results of Lamport’s demonstration.

While these types of demonstrations might be tedious and often seem to demonstrate “intuitive evidence”, they are important: first, to avoid subtle logical flaws that might be exploited by hackers; and second, for formally delimiting the trustworthiness perimeter and ensuring the validity of the model.

Knowing this perimeter allows to securely design and implement a logical model into production while addressing its vulnerabilities, which might not be only technical.

3.4. DLT Remittance applications

Ripple and Stellar are the main non-blockchain DLT technologies used in the finance sector. They are based on Byzantine consensus. Stellar forked from Ripple and both, from the onset, have been designed to target the highly regulated financial market. They naturally include features regarding “know your customer”, and anti-money laundering compliance.

Ripple targets large financial institutions with which it works to create private inter-bank blockchains to facilitate clearing and remittance. For instance, in Japan the SBI group has attracted 25 banks, representing 80% of the Japanese domestic market, to join its Ripple powered money transfer app called MoneyTap (Cook, 2019). The Ripple business model is to provide the backend system on top of which financial institutions can design consumer products.

Stellar is a Ripple fork that decided to cater to small and medium enterprises with a more social orientation. However, it follows the same Ripple strategy of providing a backbone service on top of which a company can develop consumer applications. Stellar allows anyone to deploy a server on its network, but to interoperate with other financial organizations, the entity operating the server has to be co-opted to transact with the official Stellar network.

In contrast to Stellar and Ripple, most of the blockchain based applications target end customers and intend to reduce their transaction costs. They are designed to allow end users

to transfer money as they would with any other money transfer application. They use their underlying blockchain as a transparent backend system that allows an exchange between fiat money and cryptocurrency. Examples are Etharemit (Etharemit, 2019) or Everex (Everex, 2019) that partnered with local Thailand banks to allow their users to draw remittance money from ATMs.

Other MTOs use an off-chain payment system such as: African start-up Bit-Mari (Bit-Mari, 2019) using the lightning network (LightningNetwork, 2017); the Bitcoin off-chain payment network, or, Wala (Cuen, 2018) that uses the micro-raiden (Raiden, 2019) Ethereum off-chain micro-payment network.

None of the remittance solutions reviewed address the problem of the remittance of service introduced at the beginning of this thesis.

3.5. Conclusion

The literature review showed that in public environments allowing participation of non-vetted participants, DLT applications could be used to develop an RS application. However, DLT applications expose all the codes and algorithms to the public for vulnerability analysis, mostly by hackers who would like to exploit the system. Therefore, maintaining trust in such a system is difficult but can be done in layers.

The first layer is the layer addressing the logical correctness of the application, because logical flaws that can be captured by a simple analysis of the public model are the first to be exploited. Ensuring the mathematical or logical correctness of the algorithm or process is a requirement to ensure that consensus is used in a secure environment without additional breaches in trust.

Once the logical correctness is established, the second layer is the technical aptitude of the application to deliver its service in a trustworthy manner. In a distributed computer network,

this task is done using consensus. Consensus ensures data message synchronization and security. Consensus is the cornerstone that guarantees reliable services.

Distributed Ledger Technologies have been designed to provide consensus in an untrusted environment. The blockchain, the DLT considered for this thesis, uses the proof of work algorithm as consensus. The proof of work behaves as a protective wall around Ethereum networks as it makes it extremely difficult to attack the network given that the resources necessary to rewrite Ethereum history is gigantic, and the security of the entire network augments with the number of computers joining the network.

Ethereum has computational capabilities through its Ethereum Virtual Machine (EVM). When developing in Ethereum, its resources are not free. Ethereum implements the concept of gas, which is a penalty for using EVM resources. The concept of gas encourages using exemplary coding practices to makes hacking difficult and costly, as the hacker will have to disburse an amount of money proportional to the scale of their attack.

While technically DLT has the potential to address RS, no commercial application using DLT addresses it in the way it has been described in the presentation of this research. Therefore, the need to build a model that addresses RS using DLT technology exists. That aim will be discussed in the next chapter.

CHAPTER 4

MODELING OF THE REMITTANCE OF SERVICE

As seen in the literature review, DLT are public environments where all the code, algorithms and models of operations are public. Because of that, it is important to ensure that the models and the algorithms powering DLT applications are logically correct. Otherwise, the flaws in the algorithms are the first to be exploited by hackers.

Usually the analysis addresses two steps: the first analyses the logical or mathematical correctness of the model algorithm and the second addresses the implementation vulnerabilities of the model with respect to a specific language.

The first step provides the high level correctness of the operating model, defining the model of operation, its associated algorithm and the conditions under which the entire model ensures correctness. To prove correctness, we will use an approach for solving the Byzantine problem (Lamport et al., 1982). This is the subject of this chapter.

The second step reviews the implementation of the model with a specific programing language, in this case Solidity, and ensures that the model is well implemented and addresses the known vulnerabilities of the programming language, like Solidity cybil or replay attack. This aspect of development in Solidity will be reviewed in the prototype validation chapter.

4.1. Defining the problem

The traditional MTO model for a remittance is described in Figure 4.1 and its corresponding algorithm is described in Figure 4.2.

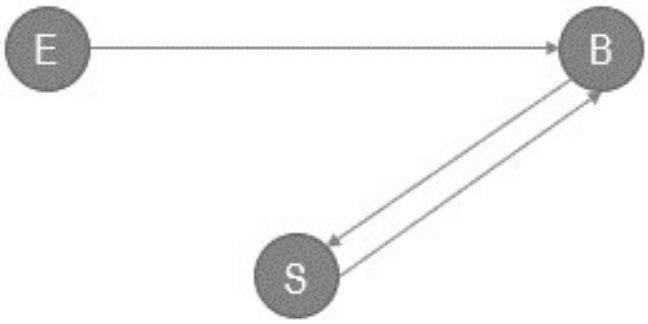


Figure 4.1: Remittance of money graph

In Figure 4.2, all steps performed in the MTO remittance model are in red and those performed by users of the remittance model are in green.

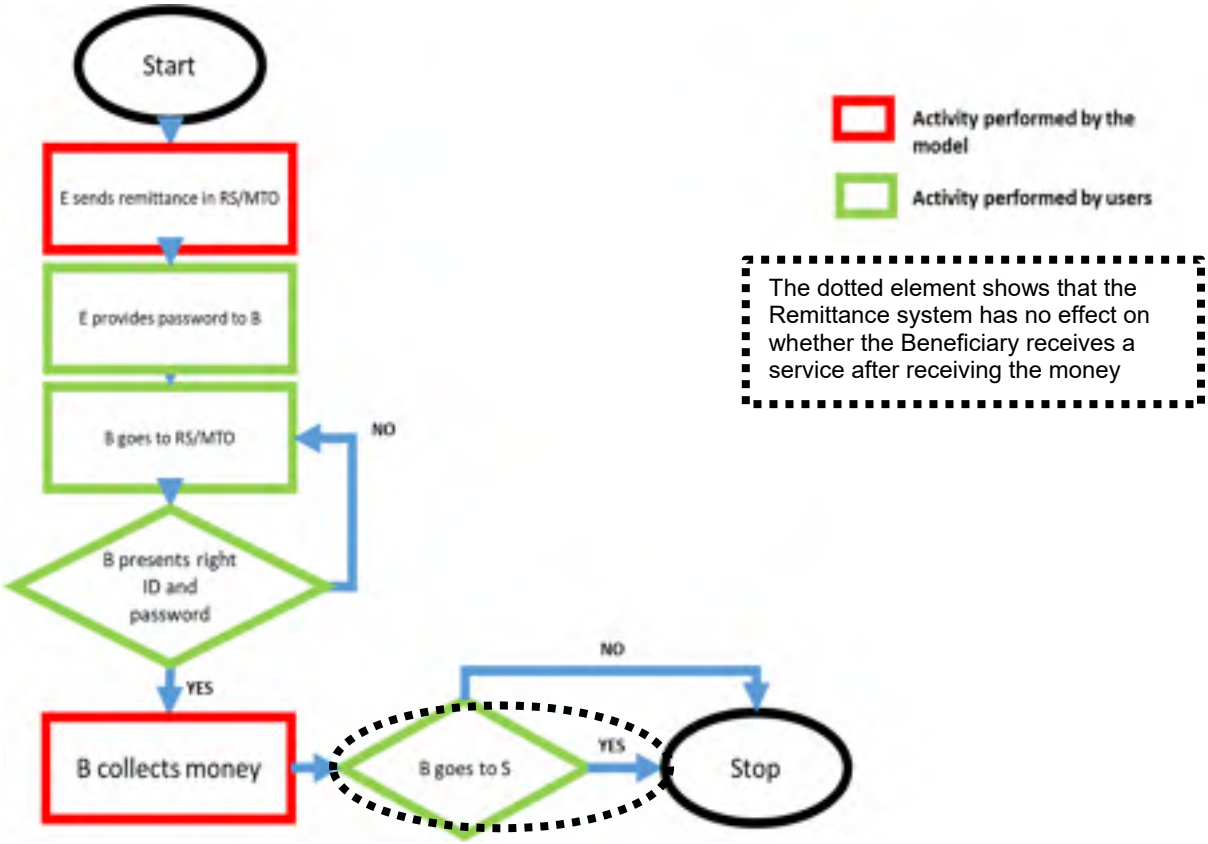


Figure 4.2: Algorithm of an MTO remittance

In this model, the role of the MTO is to channel the money from (E) to (B) and its role ends with (B) collecting the money. The MTO is not involved in the transaction between (B) and (S) and as such its role can be omitted in the overall model.

One can define an application C expressing the trustworthiness of a participant with respect to (E)'s intention. The function is equal to 1 when the participant is trustworthy, which means that he behaves as intended by (E), or 0 when he deceives (E). For instance, when (B) goes to (S) to collect a service C(B)=1, and when (S) refuses to provide a service to (B) then C(S)=0.

The trustworthiness of the entire system given by C(E, B, S) indicates whether the combined action of all the participants involved in the transaction fulfills (E)'s intention or whether the combined action of all the participants results in a system that breaks the confidence that (E) has put into the system.

$$C(E, B, S) = C(E) \wedge C(B) \wedge C(S)$$

C(E, B, S) equals 1 if the system does not allow (E) to be deceived.

By definition:

1. (E) is not deceived if: 1) (E)'s intentions are met, (B) receives a service from (S); or 2) (E) recovers his money in the case where, for whatever reason, (S) does not provide the service to (B) in the required time that we will call "dt".

2. (E) is always trustworthy since he always initiates the transaction with a clear intention when he approaches the remittance operation; therefore C(E) is always 1.

The function C is defined as:

$$\boldsymbol{C}: \boldsymbol{U} \rightarrow \{\mathbf{0}, \mathbf{1}\}$$
$$\boldsymbol{U} = \{\boldsymbol{E}, \boldsymbol{B}, \boldsymbol{S}\}$$

$$x \mapsto C(x) = \begin{cases} 1 \\ 0 \end{cases}$$

Using function C, Table 4.1 is the logical table of the remittance system described in Figure 4.1 above.

Table 4.1: Logical table of a Remittance system

	$C(E)$	$C(B)$	$C(S)$	$C(E, B, S)$
Case 1	1	0	1	0
Case 2	1	1	0	0
Case 3	1	1	1	1

In 2/3 of the cases (rows colored in grey) the transactions do not execute according to the intentions of (E). Either (B) will not go to (S) for the service, or (S) after being paid by (B) will not provide the service. In both cases, (E) will have lost his money and the intention of his transaction is not fulfilled.

It is clear that in the classical system, the fact that (E) respects his word and sends the funds does not presage anything about the behaviors of (B) and (S). The participants (nodes) of the system are totally independent: the action of one participant does not influence another node to behave in any way. This reflects a system where trust amongst the participants is absent.

4.2. Proposed solution

The remittance of service model proposed by this research is described in Figure 4.3 and comprises 4 participants: (E), (S), (B) and the trusted remittance of service application (RA). In this case, RA, which can be equated to the MTO of the classical system, plays an integral role and guarantees that the entire transaction completes successfully from beginning to end.

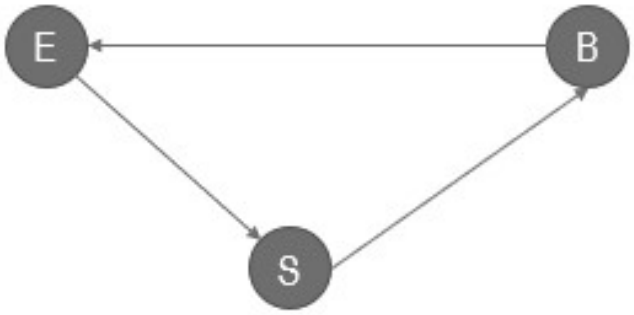


Figure 4.3: Trusted remittance of service model graph

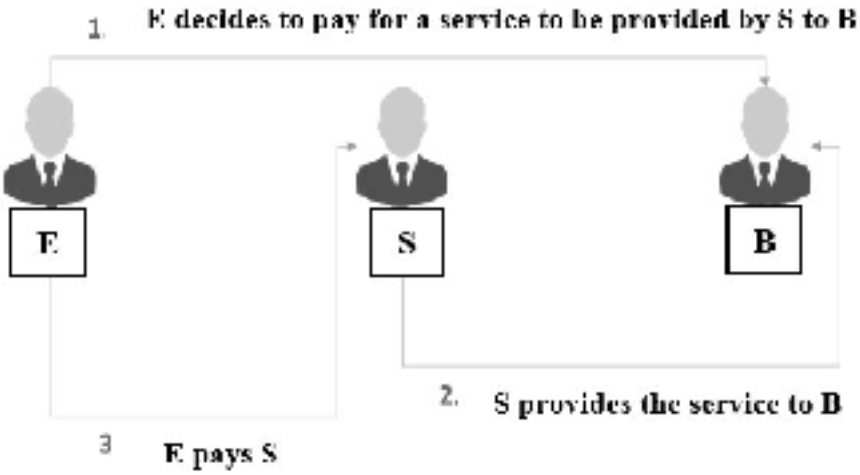


Figure 4.4: Trusted Remittance of Service model

The operating model of (RA) is defined in Figure 4.4 and its respective algorithm is described in Figure 4.5. An expeditor (E) orders (RA) to pay a service provider (S) if and only if (S) provides a service to a beneficiary (B) within a period of time “dt” and (B) confirms the service reception in (RA).

The (RA) will not be able to force (B) to go to (S). But the (RA) guarantees that if (B) does go to (S), and (B) receives a service from (S) within the determined time dt, then (S) will be paid with the money from (E) after receiving the confirmation from (B).

Figure 4.5: Proposed trusted remittance of service algorithm

The logical table of the function C for this model is provided in Table 4.2. The table shows that a system strictly adhering to the algorithm described in (Figure 4.5) will always behave in a trustworthy way in regards to (E), and C(E,B,S,RA) will be equal to 1. This result should, however, be put into perspective, as this statement is not always true. The next section examines the conditions for which this statement does hold true.

Table 4.2: Logical table of a trusted remittance of service

	$C(E)$	$C(S)$	$C(B)$	$C(E,S,B)$	Observation
Case 1	1	1	0	1	Return of money to E
Case 2	1	0	1	1	Return of money to E
Case 3	1	0	0	1	Return of money to E
Case 4	1	1	1	1	S services B

4.3. Trust risk analysis when implementing the proposed model

This section examines the conditions that would guarantee the trustworthiness perimeter of validity of the model when it is implemented. To achieve that, this research conceived an original trust risk analysis matrix that enumerates the trust risks that users face when interacting with each other. It shows the relationships between ***E***, ***B***, ***S*** and ***RA***.

When analyzing the risks of breaching the model's implementation trust with the trust risk analysis matrix, one can see that out of the 12 risks or conditions faced by users of the system, some are symmetric and thus they can be reduced to 8 conditions as listed in Table 4.3. Risks ***E-RA*** and ***S-RA***, ***B-B*** and ***S-S***, ***B-S*** and ***S-B, B-RA*** and ***RA-RA*** are symmetric. When removing these symmetric relationships, 8 conditions have to be addressed by the model if it is to behave in a trustworthy manner.

Table 4.3 Trust risk analysis matrix

	E	**B**	**S**	**RA**
E	1	Will (B) go to (S) to receive his service and notify (RA) when the service is received? In other words, can we trust (B)'s honesty? R3 1	Will (S) provide the service with the quality expected by (E, B)? In other words, can we trust (S) 's skills? R7 2	Will (RA) pay (S) when (B) receives his service? In other words, can (RA) be trusted to not repudiate its promise to pay once it receives the order from (E) and a confirmation from (B)? R1 3
B	1	Will (B) actually be the person indicated by (E)? In other words, can we trust the proof of identity presented by (B)? (unambiguous identification of participants) R2 4	Will S provide the service to B when B presents himself to S? Can S abide by the rules of RA R 5	Will (RA) be reliable and not break in the middle of a transaction? R9 6
S	1	Will (B) notify (RA) when he receives a service and if he is not happy with the quality of the received service, can we trust (B)'s evaluation of the service received by (S)? (B qualification, reputation) R4 5	Is (S) the right service provider chosen by (E)? In other words, can we trust the proof of identity presented by (S)? (unambiguous identification of participants) R2 4	Will (RA) pay (S) when (B) receives his service? In other words, can (RA) be trusted to not repudiate its promise to pay once it receives the order from (E) and a confirmation from (B)? R1 3
RA	1	Will (B) actually be the person indicated by (E)? In other words, can we trust the proof of identity presented by (B)? (unambiguous identification of participants) R2 7	Did (S) really provide the service to (B) as confirmed by (B)? In other words can we trust (B,S) confirmation of received service? (risk of collusion) R6 8	Is (RA) able to enforce all the rules and ensure that participants can engage with confidence? In other words can (RA) be trusted? R8 6

These risks are:

1) The participant must use the model for each service placed or received;
2) The service providers are vetted and their services are of good quality;
3) The model ensures that no participant is deceived when entering a transaction;
4) Each participant has unique credentials;

5) The service provider cannot refuse to service a beneficiary and the beneficiary must acknowledge the services they receive in the application implementing the model, even when they are not happy with the service received;
6) The system implementing the model is defect free, reliable and does not break under heavy load;
7) The participants do not share credentials with third parties;
8) Beneficiaries and service providers do not collude to confirm services not received or provided.

These risks introduce the limitations of (RA) that are then further analyzed through the conditions of validity of the system presented in the next section.

4.4. Perimeter of validity when implementing the proposed model

From the risk analysis, one can see that the 8 distinct risks or conditions of the proposed model can be divided in two: 1) conditions that have to be guaranteed by the system; and 2) conditions that depend on the behavior of the participants.

4.4.1. Conditions guaranteed by the system

4.4.1.1. Condition 1: The system is defect free, reliable and does not break under heavy loads

This condition ensures that the system is reliable, and can deliver service under extreme circumstances. It ensures that the trustor can enter a transaction to the trusted system with the firm conviction that the transaction will be completed as expected.

It is impossible to totally guarantee condition 1, as a totally reliable system without default does not exist. This condition will be supposed to have been met in most of the cases when the developer ensures proper quality assurance and frequently reviews the system design based on new detected threats.

4.4.1.2. Condition 2: The system ensures that no participant is deceived when entering a transaction

This condition ensures that the service that the system delivers is well implemented and provided as advertised, without any logical or implementation error.

Condition 2 is guaranteed by the model of trusted remittance of service algorithm proposed in section 4.2 where C(E,B,S) always equals 1.

4.4.1.3. Condition 3: Each system participant has credentials that uniquely identifies them and identifies their transactions

This condition ensures that every participant and the transactions he issues in the system are uniquely identified and that the participant's identity cannot be spoofed. This ensures non-repudiation of the action performed in the system.

Condition 3 requires a mechanism to uniquely identify participants of the system and the messages they exchange. This can be considered a "consensus" problem, where the 4 nodes (E), (B), (S), and (RA) exchange messages for a consensus that meets E's intention while avoiding attackers who may maliciously intercept and forge messages.

Lamport et al. (Lamport et al., 1982) demonstrated that a system of 3 or more nodes can achieve consensus in an untrusted environment if, and only if, the system uses signatures to uniquely identify participants and their messages. Therefore, to satisfy condition 3, the system must use a cryptographic signature mechanism all of the time.

4.4.2. Conditions guaranteed by the good faith of participants

4.4.2.1. Condition 4: (B) and (S) do not collude to confirm services not received or provided

This is an important condition. The beneficiary (B) and the service provider (S) must not mutually agree to confirm services that are not provided.

The consequence of condition 4 is that (RA) tolerates only 1 dishonest participant out of the 3 (E, B or S). Given that by definition (E) is always honest, if the two participants (B) and (S) are dishonest at the same time, the system no longer guarantees trust for all participants.

This condition can be restated as the participants to the system must be honest, and that is a condition that is extremely difficult to enforce technically. All computer systems are vulnerable to the human factor. As such this can be seen as a systemic condition that applies to all human made computers.

4.4.2.2. Condition 5: Each participant does not share credentials with third parties

It should be clear that the system identification mechanism will lose all of its relevance if it is not respected by the participants of the system. Therefore, it is important that participants and the system implementing the model keep the participant's identification credentials private and unique.

4.4.2.3. Condition 6: Each participant notifies the system through confirmations for each service placed or received in the transaction

The system will need confirmation for all actions performed outside of the system in real life. The system must mirror real life activity, otherwise it will be unable to fulfill its objective and participants could not trust it.

4.4.2.4. Condition 7: The service providers are vetted and their services are of good quality

This is necessary to lower the level of complaints that could lead a beneficiary (B) to refuse to acknowledge service in the system and thus prevent the service provider (S) from receiving payment. The accumulation of such grievances could lead participants to distrust the system for reasons beyond the technical capacities of the system.

Therefore it should be clear from the beginning that service providers partaking in the system are all deemed competent and are all legally licensed to provide their services.

4.4.2.5. Condition 8: Once (S) services (B), (B) agrees to confirm in (RA) that the service has been provided irrespective of his appreciation of the quality of the service

This condition is the corollary of condition 7. It must be clear from the beginning that once a beneficiary receives a service, he must acknowledge it in (RA) even when he is not satisfied by the received service. Therefore, the beneficiary should have other avenues outside of (RA) to raise grievances regarding the services received. This could be, for instance, the public consumers' bureau or the medical association in the case of a medical service.

4.4.3. RA validity perimeter

Of all of the reviewed conditions, we see that conditions 5 to 8 presuppose that the participants understand the rules of the (RA) and that the (RA) is provided under legal guidelines that provide participants other avenues to raise their grievances for actions not related to technical aspects.

For this research, these conditions are assumed to be fulfilled. The operator deciding to implement the RA model should provide the legal rules of operations that avoid any ambiguity

in the mind of the participants to the system. He might decide to only provide the technical guarantee that the system is trustworthy and let the system self-regulate through an additional peer ranking review.

Therefore for the system to be technically trustworthy and guarantee the integrity of the remittance of service transaction, it must:

1. Use a signature to uniquely identify system participants and determine the message origin;
2. Behave like a trustworthy system with respect to participant (E) as defined by the algorithm described in Figure 4.5 above.
3. The system has no more than one dishonest participant in the triple (RA, B, S)

These 3 conditions guarantee the trustworthiness of the model and constitute its perimeter of validity.

This combination of the algorithm shown in section 4.2 and the 3 conditions of validity, viewed above, constitute the Trusted Remittance of Service model.

The next section will propose an implementation model (the 3-layered stack model) for the conceptual Trusted Remittance of Service model seen above.

4.5. The 3-layered stack implementation model

Of the 3 conditions of perimeter of validity seen in the previous section, only the first 2 conditions, the use of signature and the use of the trusted remittance of service algorithm, are technical. The condition requiring not having more than one dishonest person is not technical and will not be considered in the implementation model explained in this section.

Conceptually, the Trusted Remittance of Service model proposed can be considered as the 3-layered stack described in Figure 4.6.

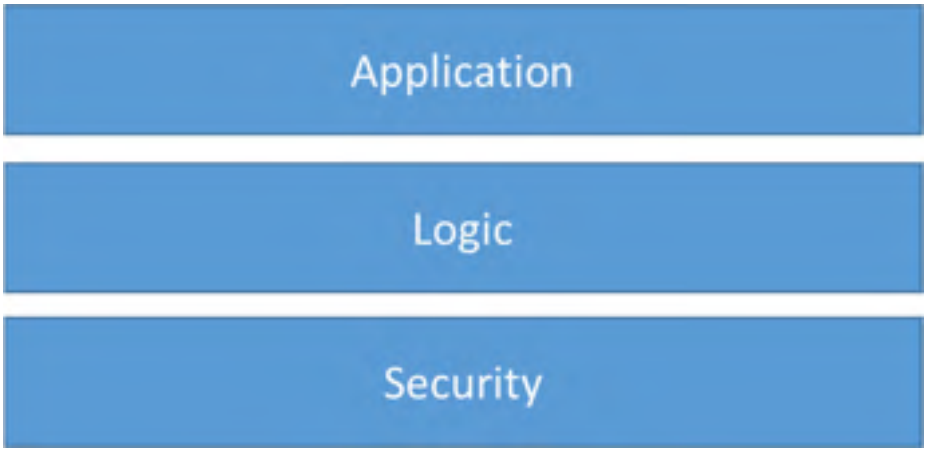


Figure 4.6: 3-layered stack model

The foundation is the security layer that contains the signature mechanism ensuring that every participant and all messages exchanged in the system are uniquely identified and safeguarded against malicious attack as explained in section 4.4.3 condition 1.

On top of this sits the remittance trustworthiness logic layer (logic layer for short), that contains the code implementing the trusted remittance of service algorithm described in Figure 4.5, Table 4.2, and in section 4.4.3 condition 2.

The top layer is the application layer that interacts with the users and exposes the lower layer functionalities to the end users. It interacts with end users through an intuitive graphical user interface and masks the internal complexity of the system to the users.

Separating the model in these layers eases the design of the application, as each layer is independent and does not need information from another layer. In this way, different technical solutions can be chosen for each layer. Layers communicate between themselves through API interfaces and each layer can be maintained independently (Richards, 2015).

It is possible to develop the entire stack from scratch with any development language such as C, Java, or Python. However, DLT systems offer some advantages when applications are viewed through the 3-layered stack lens: inherently, DLTs propose signature mechanisms that

can be considered a security layer. This is a considerable advantage because developing a system that uniformly signs every transaction and that scales globally is not trivial. In fact, the Bitcoin blockchain, the first DLT to enter production, took 10 years to go from theory to a functioning application.

Also, certain DLTs offer a smart contract development language that eases the implementation of the trustworthiness algorithm. As such, smart contracts can be considered as the logic layer where the trusted remittance algorithm is implemented.

Consequently, DLTs systems offer the foundational security layer of the remittance stack and provide the environment to implement the logic layer that naturally communicates with the security layer; this releases the developer from developing these 2 layers as well as the API that makes them communicate. Therefore by using a DLT system, one can focus on the implementation of the algorithm in the logic layer, the development of the application layer and the creation of the API interface between the logic and the application layer.

4.6. Conclusion

This chapter proposed the Trusted Remittance of Service model constituted of the trusted remittance of service algorithm, and the 3 conditions necessary for the algorithm to be implemented in a trustworthy manner.

It also proposed an implementation model: the 3-layered stack model. The two lower levels encapsulate the security and logic layers (each addressing the two first conditions of the Trusted Remittance of Service model) and the upper stack, the application layer, which exposes the inner layers to end users and which is totally up to the developer to devise.

The 3-layered stack is easily implementable using DLT. Implementation of a chosen DLT depends on the business requirements and technical specifications of the intended application. Certain DLTs possess features, or performance, that better suit the intended application and

ease the development process while increasing the application's reliability. This can maintain an end user's trust in the application.

This chapter concludes the theoretical work of the model and the next chapter provides a proof of concept implementing the 3-layered stack model of the Trusted Remittance of Service model into a prototype.

CHAPTER 5

PROOF OF CONCEPT

5.1. Objectives

We start by defining the objective of the proof of concept exercise. The objective is to implement the Trusted Remittance of Service model into a prototype, simulate a real life trusted remittance of service transaction operation using the prototype and confirm the prototype trustworthiness, i.e. that the prototype will not deceive its users.

The trusted remittance of service transactions will consist of an emitter (or expeditor), a beneficiary, a service provider and the trusted remittance of service application which does not use a third party to validate the transaction. Every transaction starts with the emitter logging into the system and choosing a beneficiary, a service provider and sending the money. The service provider and the beneficiary are notified of the availability of funds and the beneficiary goes to receive the service from the service provider. The beneficiary confirms that he received the service in the application. The service provider confirms that he provided the service. The prototype sends the money to the service provider, or in the case that a confirmation is not received after 1 hour from the beginning of the transaction, the emitter gets his money back.

To ensure the prototype's trustworthiness, the prototype must:

1) Guarantee that the validity conditions of implementation are met: i) use signatures, ii) use the Trusted Remittance of Service algorithm and iii) do not have more than one dishonest person; and

2) Reliably provide trusted remittance of service transactions in correctly implementing the 3-layered stack implementation model. For that, each layer must be tested. As the trustworthiness of the entire prototype is compromised when one layer is compromised.

5.2. Conditions of success

The proof of concept will be considered as a success if:

1) The prototype correctly implements the 3-layered stack model and passes offline tests guaranteeing that the model has been implemented correctly and that it is reliable and secure;

2) The prototype can reliably perform 100 trusted remittance transactions in the Ethereum public test ledger that is the closest to a real life operation condition.

5.3. Methodology of the proof of concept

The proof of concept will follow the following steps:

1) Choose the right Distributed Ledger Technology to implement the trusted remittance of service application;
2) Design the experimental prototype, detailed in Chapter 6;
3) Define the tests necessary to validate the prototype based on the objectives and conditions of success of the proof of concept, detailed in Chapter 7 ;
4) Test the prototype through simulation, also detailed in Chapter 7;
5) Draw conclusions, detailed in part in Chapter 7 and in the Conclusion.

5.4. Choosing the DLT for the software prototype

This section uses the Kaiwen DLT decision tree (Kaiwen Zhang, 2019) to choose the best DLT for our prototype. The Kaiwen DLT decision tree is a simple, intuitive, yet powerful tool for choosing a DLT proposed by Professor Kaiwen Zhang during his seminar on the blockchain at the École de Technologie Supérieure in 2019.

The objective of the proof of concept consists in offering a remittance of service application (RA) that provides trusted remittance of services (TRS) as already described in Figure 4.4 to any triple willing to participate in a TRS transaction (Expeditor, Beneficiary, Service Provider).

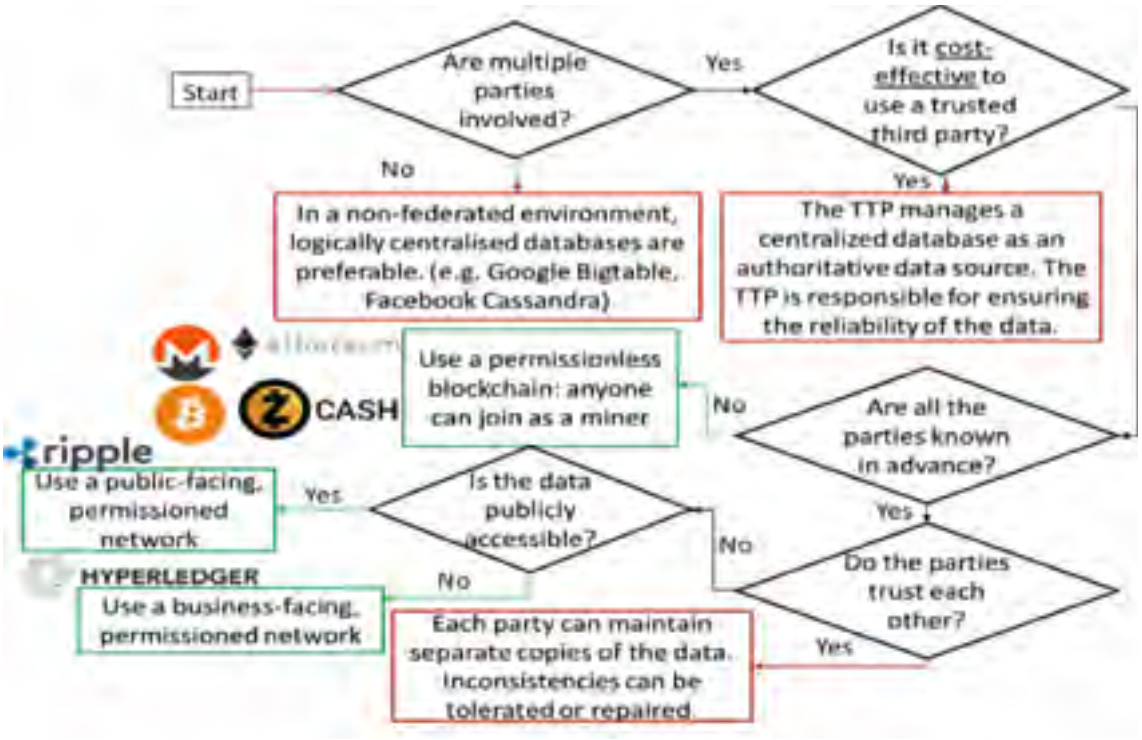


Figure 5.1: Kaiwen DLT tree

Participants to the TRS transaction can be located anywhere in the world and should not be bound by any country or third party limitation. The RA application should be able to scale globally and accept new participants while providing acceptable transaction latency.

This objective will be analyzed through the Kaiwen DLT decision tree to choose the best suitable DLT.

5.4.1. Proof of concept objective analysis using the Kaiwen DLT decision tree

This analysis consists of responding to the five questions of the Kaiwen DLT decision tree to determine the most suitable DLT based on the needs of the proof of concept.

5.4.1.1. Are multiple participants involved?

Multiple participants are involved and are described below.

The remittance service application (RA) operator: He manages the technical infrastructure, deploys the smart contract in the DLT and is in charge of ensuring that all the computer systems and smart contracts work as expected.

Money sender or Expeditor (E): Families, friends, and investors living in countries where they have access to funds and access to the (RA) for sending money to beneficiaries located in foreign countries.

Beneficiary (B): Friends, partners and family who benefit from receiving either money or a service (e.g., education, medical).

Third party Service Provider (S): Schools, universities, hospitals, shops, or other service providers who connect with the (RA) and receive payment for providing a service to the beneficiary.

5.4.1.2. Is it cost effective or suitable to use a trusted third party?

In our research and as required by the objective of the proof of concept, using a third party would introduce an exaggerated level of risk.

5.4.1.3. Are all the parties known in advance?

Anyone can participate in the remittance of service transactions and participants are not known in advance.

5.4.1.4. Do the parties trust each other?

It is assumed that the parties do not trust each other. Malicious participants may exist.

5.4.1.5. Is the data publicly available?

The DLT stores account balances, smart contracts and transaction history are all publicly available.

5.4.2. Choice of the DLT

Using our answers and plugging them into the Kaiwen DLT decision tree (Figure 5.1), the blockchain family (Bitcoin, Ethereum, Alt-coin) appears as the best-suited technology.

Within the blockchain family we chose Ethereum because it proposes smart contract development possibilities and it is the second largest blockchain after Bitcoin. Its security layer has been tested for professional activities for years.

When we consider the other DLT available, Hyperledger Fabric (Androulaki et al., 2018) and Ripple (Ripple, 2017) families, a closer look at the proof of concept objectives eliminates them. Both restrict access to their network. The operators of both networks decide who participates in the network. As such, the operator acts as a trusted third party entity. This goes against of the requirements of the proof of concept objective. Both families of DLT are geared toward business companies, mostly financial and logistic entities, needing a level of control and privacy over their data.

5.5. Conclusion

This chapter described the process to set the objectives and requirements for our proof of concept and choose the right DLT. At this point, all the information to start designing and coding the prototype itself is available: i) the trusted remittance of service model consisting of the trusted remittance of service algorithm and its implementation perimeter of validity; ii) the objectives and conditions of success of our proof of concept and iii) the DLT system on which to implement.

The next chapter guides us through all the reflection and architectural decisions needed to produce a complete prototype.

CHAPTER 6

SOFTWARE PROTOTYPE DESIGN

The previous chapter provided the rationale for using Ethereum as the chosen DLT to be used for developing a software prototype needed to validate the model.

This chapter describes the design decisions concerning the software prototype implementing the proposed 3-layered stack model of a trusted remittance. It explains the technical decisions taken when implementing the prototype. Figure 6.1 gives an overview of the technologies and applications used to design the application.

6.1. Prototype Software Architecture Design Decisions

In order to experiment the proposed model, the 3 different layers of the remittance stack will use the following technologies in this initial experimental prototype:

- The Security layer uses Ethereum security mechanisms: Ethereum elliptic signature, Hashes and Ethereum consensus;

- The Logic layer uses Solidity for developing the smart contracts that embed the remittance trustworthiness logic;

- The Application layer will consist of four main components as described in Figure 6.4: 1) a mobile rendering module; 2) an application server; 3) a database that contains the application server data and configurations; and 4) a blockchain were transaction data are stored and executed.

Conceptually, the blockchain functionalities are split in two: 1) the security; and 2) the database functions.

Given that the relational database is used to store user information and logs and does not play any other particular role in the application, its configuration will not be detailed in this section.

Figure 6.1: Overall architecture and Technologies used

As explained in Chapter 4, there is no need to discuss the security layer, which consists of the Ethereum signing and hashing scheme, because the Ethereum (DLT) security layer provides the necessary security conditions for the prototype. However, as we are using Ethereum for implementing the trusted remittance of service model and we are using Solidity as the development language, we have to account for one of the Ethereum Solidity weaknesses, the Sybil attack. This attack duplicates signatures and allows an attacker access to the system. In this case, it is necessary to add an identity key management system. This additional security mechanism has been implemented in the application layer.

The next section focuses on the logic layer detailing the smart contract and the application layer that exposes all the inner layers to the user.

6.2. The logic layer

The smart contract is implemented in Solidity, a low level language prone to logical defects and security vulnerabilities similar to those of the C programming language. Consequently it needs thorough testing before securely launching into production. Solidity provides a limited set of primitives to develop complex data structures. That makes developing using Solidity counter intuitive for software engineers used to higher level programming languages such as Python for instance. For this prototype, data are stored in a relational database structure that had to be entirely designed.

Other programming languages could have been chosen for developing the smart contracts in Ethereum: Low Level Lysp (https://lll-docs.readthedocs.io/en/latest/lll_introduction.html) or Vyper (https://github.com/ethereum/vyper). However, neither has reached the level of maturity of Solidity, which is the most supported Ethereum language.

The smart contract proposed in this thesis implements logic that removes any risk of (E) defaulting on paying (S). (S) can trust that by using a DLT remittance service, he or she will always be paid by (E) after providing a service to (B). (E) knows that when entering a DLT transaction, there will be no way to cancel payment to (S) once there is confirmation that (B) has been provided service.

The smart contract will hold the money from the expeditor until a signed confirmation of the service provided is received from both the beneficiary and the service provider. In the case where no confirmation is received after a time dt, set to one hour for the test, the money is sent back to the expeditor.

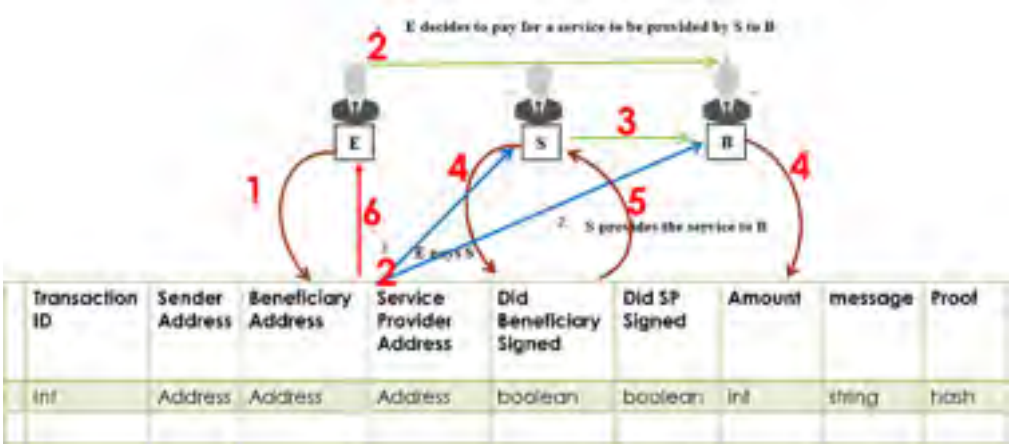


Figure 6.2: Smart Contract DB design

The smart contract shown in Figure 6.2 will behave as follows:

1. The Expeditor chooses the beneficiary, the service provider, the amount of money available for the service provider, and a secret password and writes this information in the smart contract. From that moment, the transaction is active for one hour (this time may be changed);

2. The Expeditor sends the password to the beneficiary off-chain. The application layer sends a notification to the beneficiary and the service provider that a transaction is available for them;

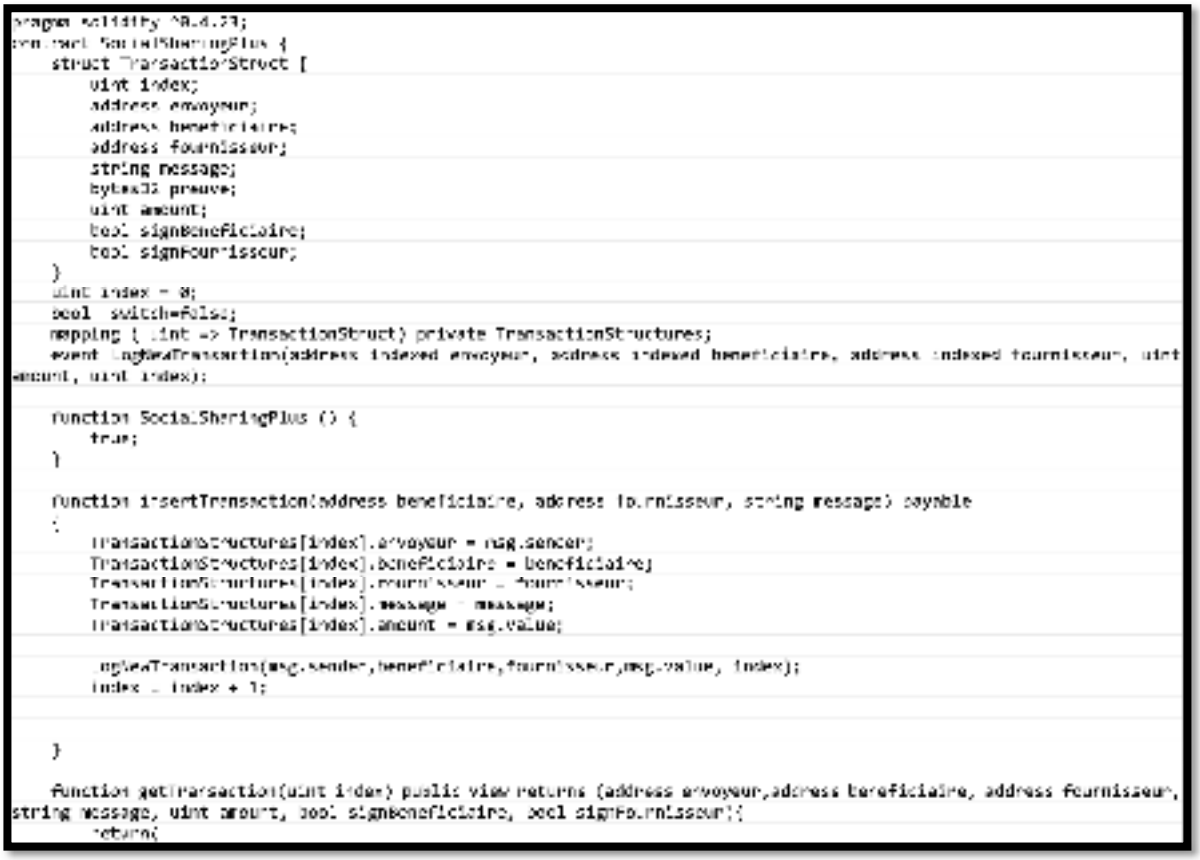
3. The Beneficiary goes to the service provider to get the service;

4. The Beneficiary and the service provider confirm that the service has been provided in the smart contract;

5. The smart contract sends the money to the service provider;

6. If the smart contract does not receive all the signed confirmation messages before the end of one hour, the smart contract returns the money to the Expeditor.

It must be noted that the smart contract provides all these functionalities through an API that the application layer uses to interact with it. A snippet of the smart contract code is provided in Figure 6.3.



```
pragma solidity ^0.4.24;
contract SocialSharingPlus {
    struct TransactionStruct {
        uint index;
        address envoyeur;
        address beneficiaire;
        address fournisseur;
        string message;
        bytes32 preuve;
        uint amount;
        bool signBeneficiaire;
        bool signFournisseur;
    }
    uint index = 0;
    bool  switch=false;
    mapping ( uint => TransactionStruct) private TransactionStructures;
    event LogNewTransaction(address indexed envoyeur, address indexed beneficiaire, address indexed fournisseur, uint
amount, uint index);

    function SocialSharingPlus () {
        true;
    }

    function insertTransaction(address beneficiaire, address fournisseur, string message) payable
    {
        TransactionStructures[index].envoyeur = msg.sender;
        TransactionStructures[index].beneficiaire = beneficiaire;
        TransactionStructures[index].fournisseur = fournisseur;
        TransactionStructures[index].message = message;
        TransactionStructures[index].amount = msg.value;

        LogNewTransaction(msg.sender,beneficiaire,fournisseur,msg.value, index);
        index = index + 1;


    }

    function getTransaction(uint index) public view returns (address envoyeur,address beneficiaire, address fournisseur,
string message, uint amount, bool signBeneficiaire, bool signFournisseur){
        return(
```

Figure 6.3: Solidity code sample

In this first proof of concept, the possibility to connect to a financial institution using Ethereum Oracle is not implemented. All transactions are done in Ether. As public exchanges converting Ether to fiat currency exist, this functionality is not necessary in this prototype.

6.3. The application layer

The Application layer will consist of 4 main components as described in Figure 6.4: a mobile rendering module; an application server; a database that contains the application server data and configurations; and a blockchain were transaction data are stored and executed.

Figure 6.4: Application layer

Conceptually, the blockchain functionalities are split in two: security and the database functions.

Given that the SQL relational database is used to store user information and logs and does not play a particular role in the application, its configuration will not be detailed in this section.

6.3.1. Mobile rendering module

The software prototype is a mobile application developed using HTML, CSS and the Javascript bootstrap framework. It consists of a number of user interfaces (Figure 6.4) that guide the user through the functionalities and it is described in more detail below.

The bootstrap Javascript framework allowed the mobile application to be responsive and rendered correctly on any type of mobile device. The mobile application interface represented in the Figure 6.5 consists of the following twelve user interfaces:

- User interface 1: The user must log in via his email address and his password that was previously defined when registering on the platform.

- User interface 2: The successful connection gives access to the welcome window, which clearly shows the balance available on the customer's account as well as the list of service providers available on the application.

- User interface 3: On the Side bar menu, the application offers links to return to the home page, an overview of the customer’s contracts and an option “About”.

- User interface 4: By clicking on the "My contracts" option, a summary of the contracts concerning the user is displayed. We can clearly see the three categories managed by the application: Beneficiary contract in the case where the user is the recipient of a transfer of money; Sender Contract, in the case where the user is the initiator of a transfer; Provider contract, in the case where the user is a service provider registered on the blockchain.

- User interface 5: From user interface 2, a click on one of the service providers listed initiates the money transfer process by proposing, as indicated on this screen, the list of potential recipients.

- User interface 6: Once the recipient has been selected, a contract validation window is displayed that allows the user to enter a password and the amount to be sent.

- User interface 7: The confirmation of the operation initiated on user interface 6 produces the report presented on this screen and can be consulted at any time from the “My contracts” option.

- User interface 8: The application also offers the possibility of editing the user’s profile at will.

- User interface 9: A simulation of the beneficiary account is included on this screen. It shows the list of contracts for a second user. We can find the contract previously deployed at the end of the operation initiated on user interface 6.

- User interface 10: A click on one of the contracts concerning the user, and a window allowing him to consult the information relating to the transfer that has been sent to him appears. In this case, he has not yet signed, so the option is reserved for him as well as the possibility of carrying out a transaction. The signature of the contract gives him access to the effective reception of the transfer sent to him, depending on whether he must use it for a service provider or not. The case thus raised will necessarily require the signature of the supplier in question.

- User interface 11: A simulation of the supplier account is included on this screen. It shows the list of contracts for a service provider. We can find the contract previously deployed at the end of the operation initiated with user interface 6.

- User interface 12: A click on one of the contracts concerning the user, and a window allowing him to consult the information relating to the transfer that is dedicated to him appears. In this case, he has not yet signed, so the option is reserved for him. The signature of the contract gives him access to the actual receipt of the resulting transaction from the beneficiary.

A similar version of the prototype developed for the Web (Figure 6.6) uses different user interfaces than the mobile software prototype but it shares the same functionalities.

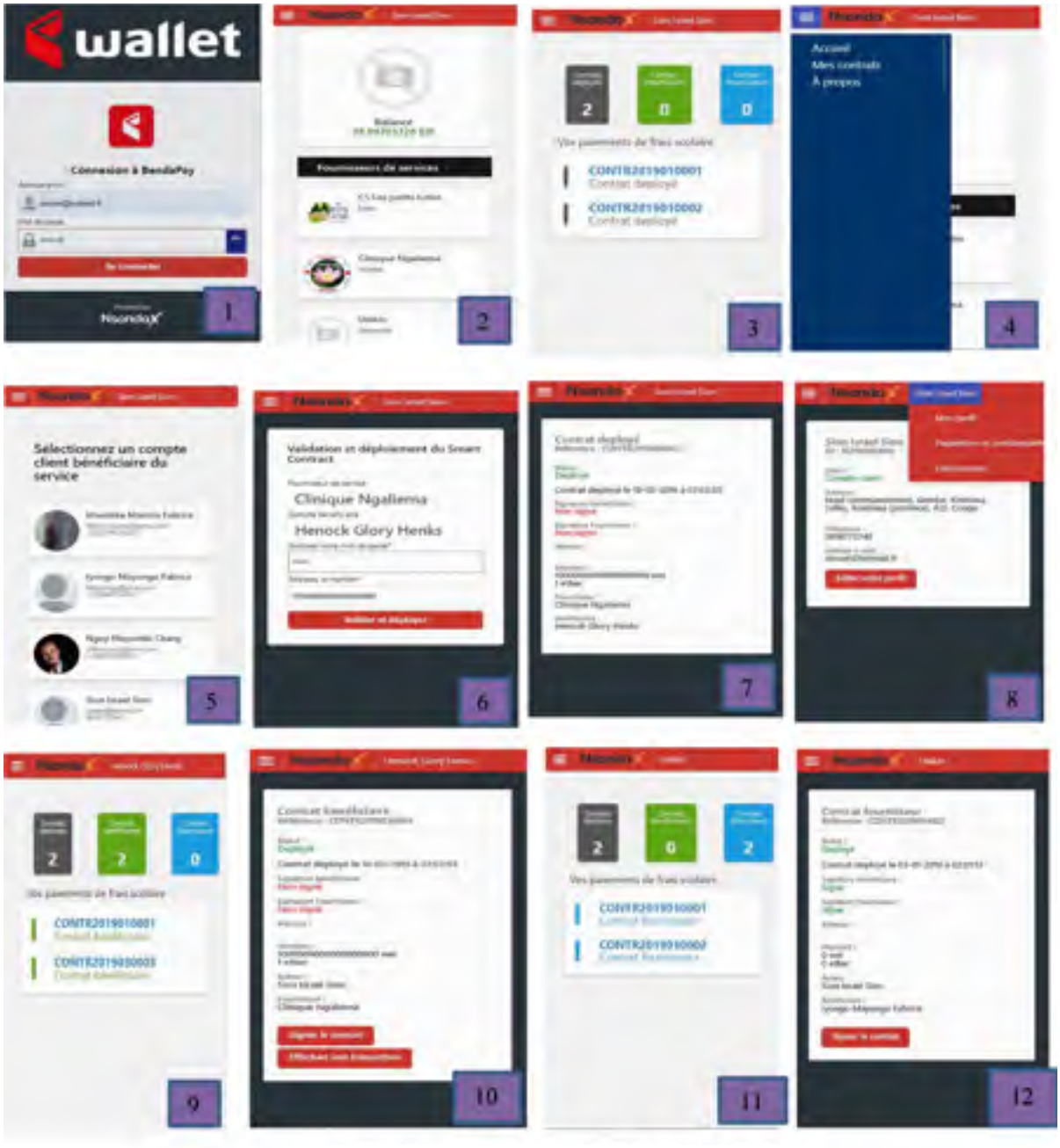


Figure 6.5: Software prototype user interfaces

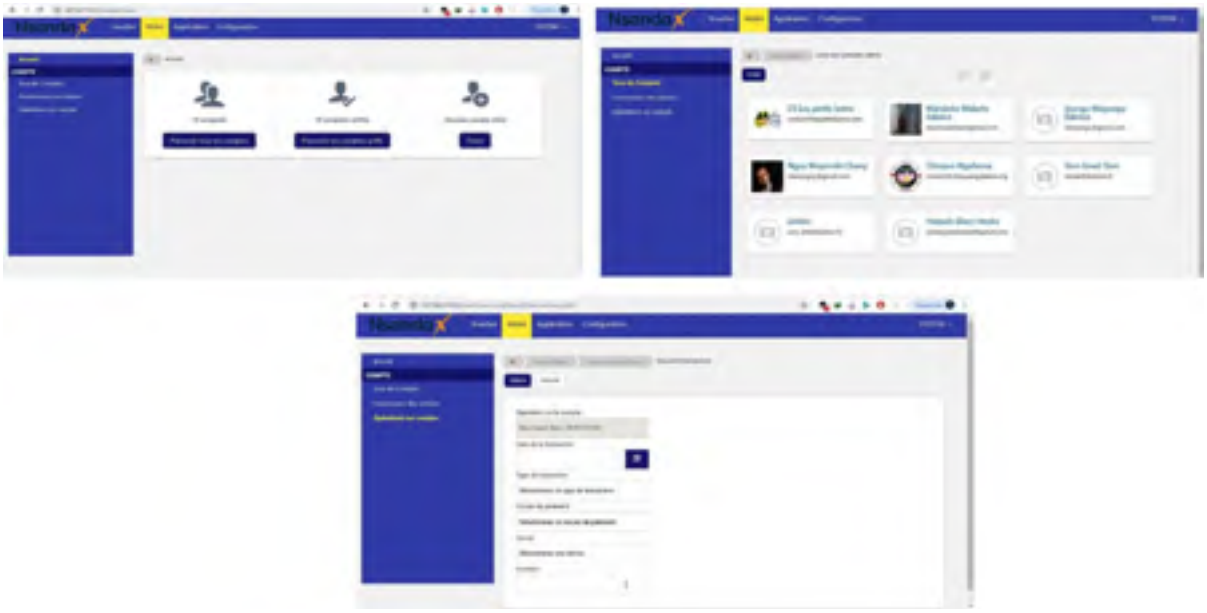


Figure 6.6: Web based interface

6.3.2. The application server

For the experimentation, an application server also had to be designed, developed and tested. This application server links the blockchain to the mobile application and it contains the application logic, not to be confused with the smart contract logic. For auditing purposes, all communication between the mobile application and the blockchain are logged in the application server database. The application server was developed using the Python Django Web development framework.

This application server communicates with the mobile client using a REST-API and it communicates with the Ethereum blockchain using the Web3.py library. The Web3 library is a library binding Ethereum blockchain REST-API to Python functions.

Other configurations could have been possible. For instance, it would have been possible to have the mobile application communicate directly with the blockchain. But since direct communication between mobile clients and the blockchain makes maintenance and support more complicated, having an intermediate application server between client and blockchain is a best practice in software engineering.

It should also be noted that with this configuration, the application server is the custodian of the private keys of all the users and acts as a key and identity management system. This configuration provides the added benefit of preventing a Sybil attack on the Ethereum network.

Using the application server as the custodian of cryptographic keys makes the entire system vulnerable in the case of an application server data breach. The fact that the cryptographic keys are stored encrypted and secured by an additional access control list reduces the possibility of compromising the system without totally eliminating it.

6.3.3. Blockchain

During development and testing, a private Ethereum blockchain was deployed using the GoEthereum client (commonly called Geth) and Ganache. Once the application was finalized, it was tested on Ropsnet.

CHAPTER 7

SIMULATION RESULTS AND ANALYSIS

This section examines whether the prototype achieves the proof of concept's objective and fulfills its conditions of success. The strategy used to ensure that the prototype correctly implements the 3-layered stack implementation model is explained after which the simulation strategy to validate the real life operation of the prototype is discussed.

7.1 Test scenario design

Before starting the real life simulation in the public Ethereum network, the prototype trustworthiness validation was done by testing 80 off line scenarios targeting the security, the logic and the application layers. This was done to ensure that the entire prototype does not lose the trust vested in it if one layer is compromised.

The list of tested scenarios is provided in Appendix 3. For each scenario, the test result was either pass or fail. Test scenarios had to simulate normal transactions and problematic transactions with the goal to clearly prove that in all cases the prototype was behaving according to the trusted remittance of service algorithm model. After preparing the scenarios, they were coded into unit tests using selenium robots to simulate users interacting on the graphical user interface of the prototype.

For the security layer, the research looked at known references of the Ethereum security vulnerabilities and tested 60 of the most dangerous that could compromise the entire system. For the logic layer, the prototype tested 12 scenarios to ensure that the trusted remittance of service algorithm was correctly implemented and for the application layer, 8 scenarios were tested.

Selenium was used to automate the interaction between the mobile application and potential users feeding information. Selenium is an application that allows the tracking of every keystroke a user enters in the system, and at each step checks the internal state of the system. Testing using Selenium not only allows for the testing of the quality of the code, but it also tests the overall behavior of the system as seen by the end user, especially the page responsiveness.

These tests were automated and repeated with different values, and after several iterations, each one correcting the previous one of all the coding defects, the final version of the prototype was subjected to all of the tests and it passed them all.

The offline tests confirmed that the prototype reliably implemented the 3-layered stack and it proved that the experimental prototype behaved according to our proposed definition of a trusted remittance of service model.

It should be stressed that, not all of the possible scenarios that exist were tested. Still, we are confident that this group of tests demonstrated a good working condition of the prototype. If we had had more time, we would have conducted more tests, especially for the logic layer dealing with the smart contract reliability. Testing smart contract reliability is an important area of research and would require much more effort than we could expend in this research.

7.2. Simulation

The simulations were performed with two substantiation objectives: first, to ensure that the prototype could operate in a trustworthy manner with respect to the intention of the expeditor on near to real life situations and second, to examine additional aspects of the prototype transactions such as processing time and financial cost when the prototype is compared to actual MTO services.

To determine real life operating conditions, the prototype was tested using Ropsten, the public Ethereum test network. Ropsten is the closest system to the real Ethereum network. Using Ropsten allows for a close approximation of how the system would behave in real life.

Simulations of trusted remittance of service transaction using the software prototype were conducted over two days (from May 22 to May 23, 2019), each from 9 am to 4 am GMT+1 using the mobile version of the prototype. 160 remittance transactions were performed. All the transaction details are publicly available on the Ropsten public log at: https://ropsten.etherscan.io/address/0x904248FE328a186CE76666ee9e2548Ab2C861336.

On the day of the transactions, the value of one Ether was $230.17 USD.

7.3. Simulation analysis

160 transactions were performed on the Ropsten network. A transaction was comprised of the stages described in Figure 7.1: 1) Every transaction starts with the emitter (or expeditor) logging into the system and choosing a beneficiary, a service provider and sending the money; 2) the service provider and the beneficiary are notified of the availability of funds and the beneficiary goes to receive the service from the service provider; 3) the beneficiary confirms that he received the service in the application; 4) the service provider confirms that he provided the service; 5) the prototype sends the money to the service provider, or in case a confirmation is not received after 1 hour from the beginning of the transaction, the emitter gets his money back.

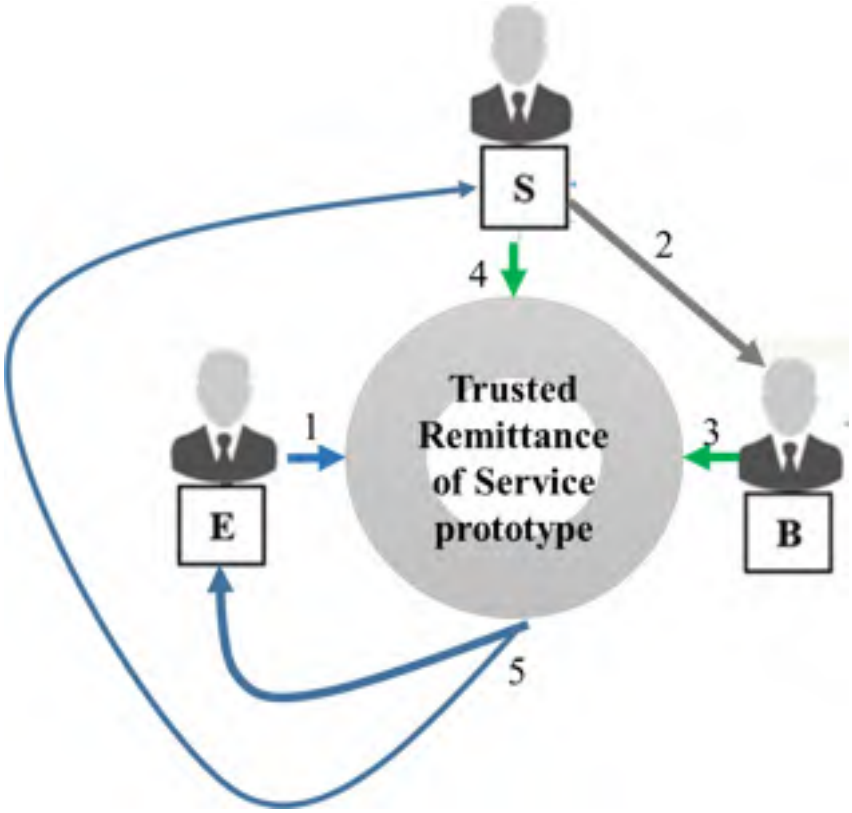


Figure 7.1: Trusted remittance of service transaction

The time to complete a command in the Ethereum network, whether for sending money or confirming a transaction, ranged from 5 seconds to 1 minute during the testing. It was observed that this duration is a function of the network load and is not related to the amount or type of commands.

The total average cost of a remittance transaction is $1.29, irrespective of the amount remitted. This is largely below the $15 that is usually paid for a remittance of $200 sent from Canada to the DRC using Western Union. Table 7.1 shows that the average cost for each step of the transaction was between 0.3 Ether to 3.08 Ether.

Table 7.1: Cost of using the application

Actor	Operation	Average cost in USD
Expeditor		
	Transferring money	0,6608527 $
Beneficiary		
	Signing Confirmation	0,21955631 $
	Choosing a service provider	0,20657076 $
Service provider		
	Signing confirmation	0,20657076 $
Total cost of the transaction		1,29355053$4

It must be emphasized that the cost we are referring to in this analysis is the gas cost, which is the cost charged by the Ethereum network to perform a transaction. As described in the literature review, every operation (command) performed in the Ethereum network costs Ether and it is identified as a gas fee. This gas fee can be understood as a constraint to oblige the developer of a smart contract to use the least resource intensive operation possible. The gas fees are distributed to miners of the Ethereum network as compensation for maintaining security.

In a commercial sense, in addition to the gas fees that are collected by the network, the operator of the smart contract should also set its own fees, covering commercial overhead and benefits. For this experimentation, the commercial remittance fee was set to zero and only the cost of using the Ethereum resource network has been studied. However, it is reasonable to think that even after adding a realistic overhead charge by the remitter of service operator, the prototype could still be commercially competitive compared to an MTO.

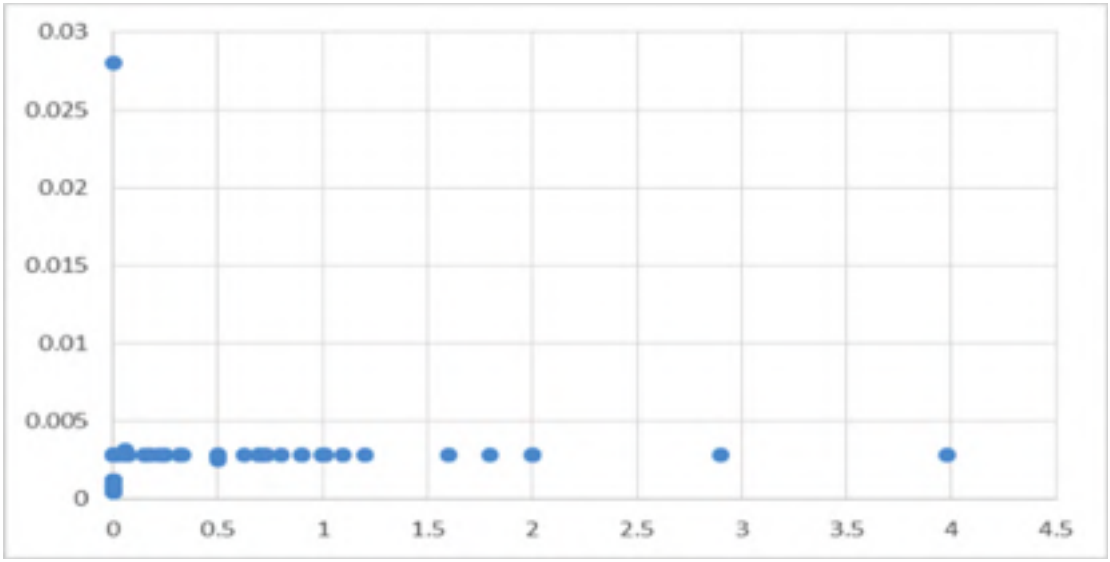


Figure 7.2: Gas Cost as a function of the amount remitted
X-axis: amount remitted in Ether, Y-axis gas cost in Ether

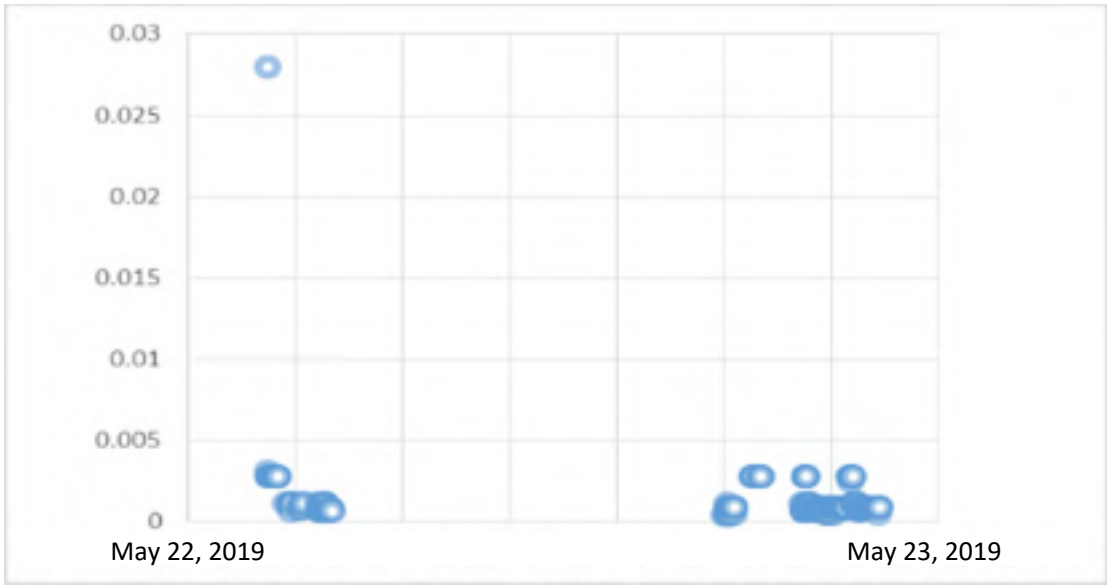


Figure 7.3: Gas cost as a function of time of the transaction
X-axis: days from May 22 to May 23, 2019, Y-axis: gas cost in Ether transactions

Analysis of the simulation also shows that the amount remitted has no influence on the cost of the transfer, i.e. the gas cost charged by the Ethereum network (see Figure 7.2). It seems that there is a slight dependence between the time a transaction is completed and the gas cost

however (see Figure 7.3). This could be linked to the load on the system at certain times during the day.

A major difference between the traditional remittance of service model and the one implemented in Ethereum is that the cost of a transaction is spread among the participants. In the current MTO model, the expeditor bears all the costs of the transaction. In the proposed remittance of service model using Ethereum, each participant has to pay a fee.

This is mandatory because there are no free transactions in the Ethereum network. This could be addressed in a future iteration of the prototype by adding functionality that computes the expected gas cost to be paid by each participant and incorporating it in the expeditor fees. In this manner, the system will automatically reimburse gas fees to the other participants.

CONCLUSION

As defined in the first chapter of this thesis, the objectives of this research were: 1) to validate quantitatively whether the observation that remittance of service was an issue of concern in the city of Kinshasa was true, 2) to develop a computer model that ensures that the expeditor of a remittance transaction has control over the intended use of the money transfer, and 3) to reduce the cost of the remittance of service and to make the remittance of service process more user friendly.

One can conclude that the research has reached its objectives. The research questions guided the methodology that consisted of starting with field research to gather data and confirming whether the perceptions during informal observations were supported by evidence. The survey results presented in Chapter 2 also enabled us to gather information on the remittance user's experience and better understand the ultimate purpose of the remittances received so that the analysis of the survey could generate useful and actionable insights.

The survey generated qualitative and quantitative data illustrating the remittance landscape in the city of Kinshasa and it confirmed that the remittance of service was indeed an issue of interest for the inhabitants of the city of Kinshasa. Of the issues raised by the survey, the research focused on addressing the problem of the remittance of service. The information collected during the survey was used to design a novel Trusted Remittance of Service conceptual model containing a Trusted Remittance of Service Algorithm and the conditions necessary to deploy it in a trustworthy manner. Additionally, an implementation model, the 3-layered stack implementation model, was proposed to guide engineers on how to implement the Trusted Remittance of Service Model. This was presented in Chapter 4.

Chapter 5 explained the proof of concept for implementing the Trusted Remittance of Service model into a prototype and testing it in conditions similar to real life conditions. Given that the model proposed in Chapter 4 is a conceptual model, additional information was needed in order to choose the technology best suited for its implementation. Chapter 5, proposed a

methodology for gathering the necessary additional information needed for applying this model based on the Kaiwen DLT decision tree.

Chapter 6 described the choice of technology and proposed the software architecture to design and implement a prototype of the model, which was experimented in iterations, evaluated and discussed in Chapter 7. The tested application was delivered through a mobile application with a user friendly graphical interface to provide remittance of service without the disagreement highlighted in the survey of the Chapter 2.

Chapter 7 evaluated the prototype application based on the objectives of the proof of concept. The evaluation concluded that the tested application guaranteed the expeditor's control over the remittance money until the service is delivered to its intended beneficiary, otherwise, the expeditor gets its money back.

The economic evaluation, although simplistic and which does not take into account real commercial overhead, has shown that the prototype has the potential to be competitive if used in commercial settings. However, further research is needed to confirm this statement beyond any reasonable doubt.

The thesis demonstrated that remittance of service could successfully be modeled using a three-layered stack for a trusted remittance of service model (TRS model). In addition, the experiment using DLT shows that it is a good candidate for a technical platform to implement this model as it ordinarily offers a security layer, which is the foundation of the stack, and it offers technical features that can easily implement the other two layers.

The proof of concept showed that it was possible to successfully implement the proposed TRS model using Ethereum, which is from the blockchain DLT family, although the TRS model is not dependent on a single set of technologies. Implementing the TRS model with a different DLT, such as Ripple or Stellar, and comparing results with the Ethereum implementation

described here could be instructive. It could show which DLT application is more suitable for developing remittance of service applications.

While this thesis has shown that developing a remittance of service application based on the TRS model is technically possible, offering the remittance service to a wider public would call for building an ecosystem for acquiring service providers and enhancing the experience of the participants. This would need business and market expertise. These aspects are not addressed in this research and could be future areas to explore.

This thesis acknowledges that although this research shows that using a blockchain system substantially reduces the absolute cost of a remitting service, other aspects such as the user experience, have not been thoroughly researched. Providing a proper analysis of the user experience and measuring how much the application actually reduces the total cost of receiving money could also be an interesting addition to this research.

APPENDIX 1

REMITTANCE DESK RESEARCH AND QUALITATIVE REVIEW

This research report introduces the African remittance business market. It explores the actors, the potential for innovation, and the existing problems while exploring how technology is currently influencing it. It also summarizes the interview conducted with the different participants of the remittance business.

A1.1 Introduction to African remittance business

Remittance is an important consequence of human migration. In 2017, more than 250 million people lived outside of their country of birth. For 2018, global remittance flows reached $689 billion USD, compared to $633 billion USD in 2017. Remittance to low- and middle-income countries reached $529 billion USD in 2018, an increase of 9.6% from the $483 billion USD spent in 2017 (WorldBank, 2019).

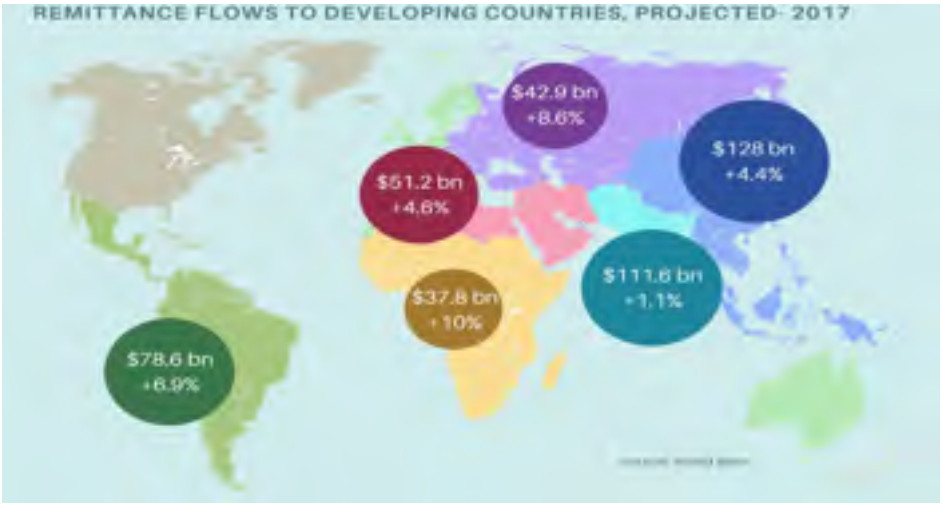


Figure A.1: Remittance flows to developing countries, projected 2017 (WorldBank, 2017)

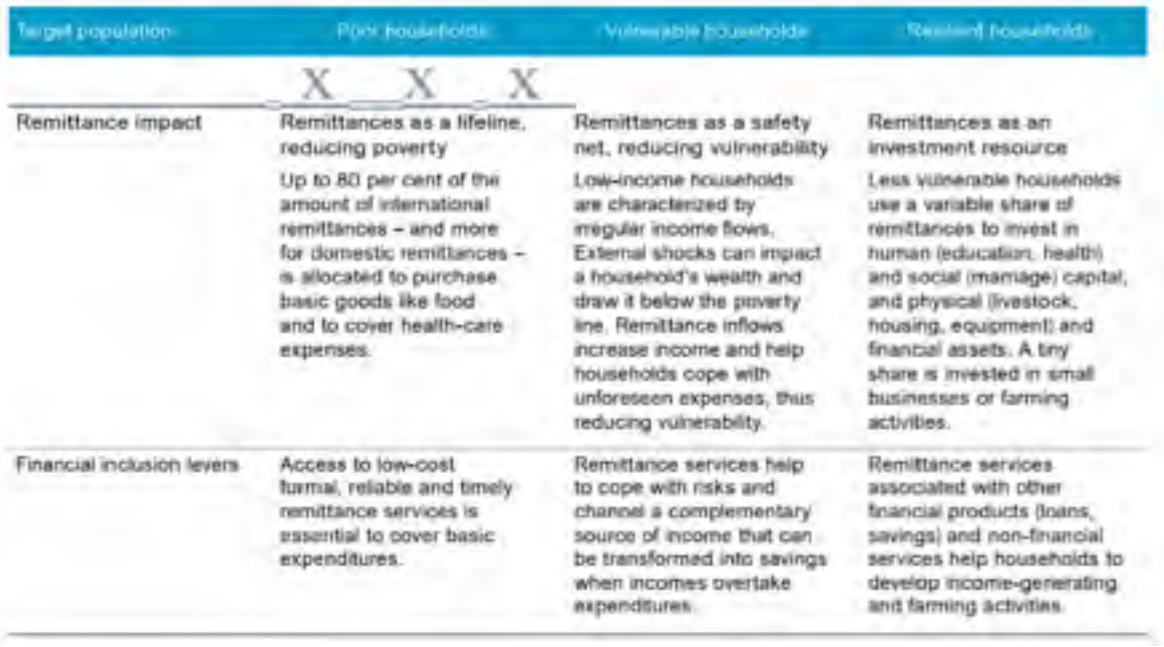
Overall, worldwide remittance is expected to grow in the future: flows to Sub-Saharan Africa were projected to increase by 10%, to Europe and Central Asia by 8.6%, and to Latin America and the Caribbean by 6.9% in 2017 (Figure A.1).

In terms of volume, remittance amounts sent back by migrants to their countries of origin account for more than the official public development assistance from all of the developed countries toward developing countries (Figure A.2).

Figure A.2: Remittance flow compared to Foreign Direct Investment and other inflow to developing countries (WorldBank, 2019)

The money sent home by immigrants provides a financial lifeline to their families. A recent World Bank study concluded that remittances are accompanied by a reduction in the number of people in poverty (Figure A.3). For example, a substantial part of remittances in Mali is saved by the recipients for unexpected events, thus serving as insurance for the entire household (Ponsot & Obegi, 2010). In rural areas of Nigeria, it has been observed that food

security improved considerably with recent increases in remittances (Babatunde & Martinetti, 2010).



Target population	Poor households	Vulnerable households	Resilient households
	X	X	X
Remittance impact	Remittances as a lifeline, reducing poverty Up to 80 per cent of the amount of international remittances – and more for domestic remittances – is allocated to purchase basic goods like food and to cover health-care expenses.	Remittances as a safety net, reducing vulnerability Low-income households are characterized by irregular income flows. External shocks can impact a household's wealth and draw it below the poverty line. Remittance inflows increase income and help households cope with unforeseen expenses, thus reducing vulnerability.	Remittances as an investment resource Less vulnerable households use a variable share of remittances to invest in human (education, health) and social (marriage) capital, and physical (livestock, housing, equipment) and financial assets. A tiny share is invested in small businesses or farming activities.
Financial inclusion levers	Access to low-cost formal, reliable and timely remittance services is essential to cover basic expenditures.	Remittance services help to cope with risks and channel a complementary source of income that can be transformed into savings when incomes overtake expenditures.	Remittance services associated with other financial products (loans, savings) and non-financial services help households to develop income-generating and farming activities.

Figure A.3: Remittance impact according to household income level (IFAD, 2016)

Recognizing the growing importance of remittance in decreasing global poverty, the UN has declared lowering the costs of remittances to less than 3% by 2030 as a sustainable development goal.

Another interesting aspect frequently raised during interviews with individuals sending money to the DRC was the lack of control over remitted funds. Eight out of 12 persons interviewed explained that they have felt abused by the usage of remittance services. One person sent remittance money for the building of a house to his brother who then pocketed the money and did not build anything. Two parents complained that the money they send for their children's school fees sometimes disappeared and they had to send the money more than once. All these observations illustrate the importance of having a trustworthy person receiving the money back at home when the remittance money is sent with a clear goal.

A1.2 Cost of remittance

Today, sub-Saharan Africa has the highest remittance costs in the world. In 2018, globally, the average cost of sending a remittance of $200 USD (i.e. inclusive of all fees and charges) was 7.2% of the transfer amount compared to 10% for Africa (WorldBank, 2019).

Remitting money within Africa is even more costly than remitting it from outside of Africa to Africa. For instance, remittances from the United Arab Emirates to Sudan or South Sudan tend to have the lowest costs. Alternatively, intraregional corridors originating in Nigeria, Angola and South Africa are among those with the most expensive remittance costs (WorldBank, 2019).

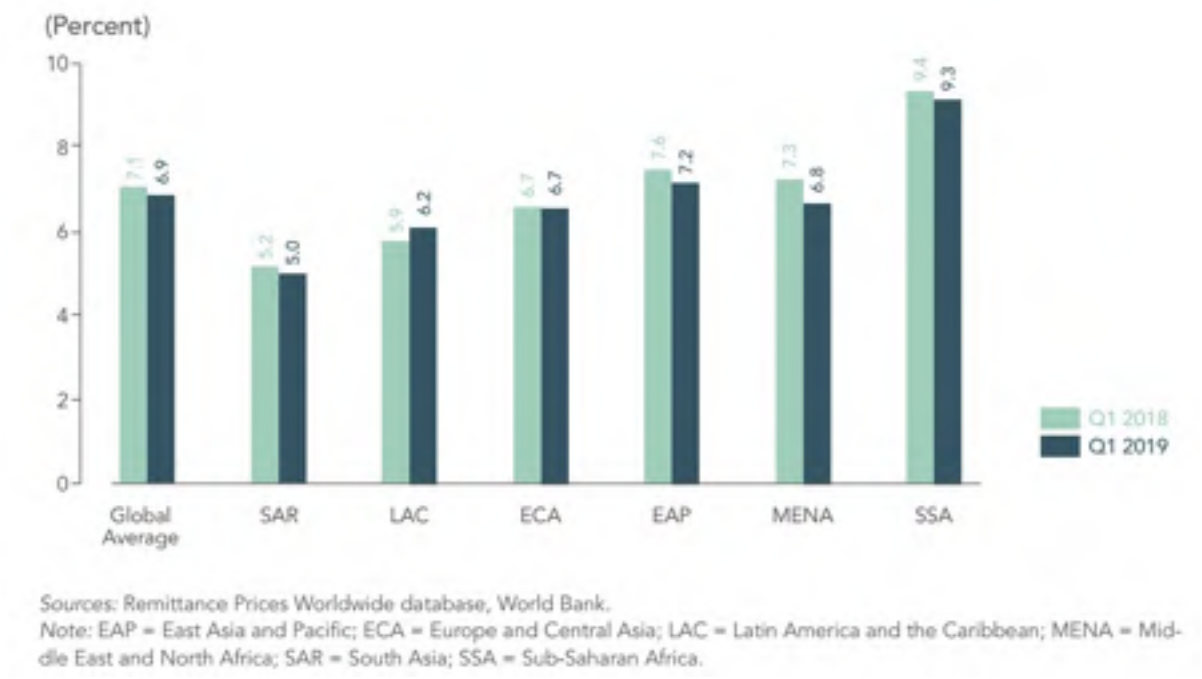


Figure A.4: Global cost of sending $200 per economic zone (WorldBank, 2019)

Remittance costs across many African corridors and small islands in the Pacific remain above 10%, because of: 1) low volumes of formal flows, 2) de-risking behavior of commercial banks,

3) inadequate penetration of new technologies, and 4) lack of a competitive market environment (WorldBank, 2019).

Of these 4 causes, de-risking behavior, technologies and lack of competition are the main drivers of high remittance costs in Africa and will be examined in the sections below.

A1.3 De-risking behavior of commercial banks

Since 2016, increasingly tighter regulations have been issued on international Money Transfer Organizations (MTO) concerning Anti-Money Laundering/Countering Financing of Terrorism (AML/CFT). Global terrorism and money laundering has developed into a major global concern toward which governments worldwide responded by tightening regulations on crossborder transfers, including remittance flow, to an extent where regulation becomes an additional barrier to the adoption of remittance services (Alani, 2017; FATF, 2015; IFAD, 2016; Nikos, 2018; WorldBank, 2017).

Compliance to AML/CFT has given rise to a "de-risking" attitude: *"the phenomenon of financial institutions terminating or restricting business relationships with clients or categories of clients to avoid, rather than manage, risk" (FATF, 2015; IFAD, 2016).* African countries, such as Somalia where incoming money is feared to fund terrorism, are the first victims of de-risking. For example, in 2014, the biggest MTO in Africa—Dahabshiil, founded in Somaliland—which from its headquarters in Dubai transacts in over 24,000 outlets and employs more than 2,000 people across 126 countries, had its bank account closed overnight by Barclays. Dahabshiil had to sue Barclays for wrongful termination of service in order to have its account temporarily re-opened pending a formal trial (IFAD, 2016).

Banks justify de-risking in view of the heavy burden involved in abiding to AML/CFT, and the possible severe penalties they might incur if they are found to be contravening it. Banks have to archive all of their transactions and be available to respond to requests from multiple regulators in multiple countries, in an environment where regulatory requirements vary among global, regional, and local regulatory bodies. Managing this situation obliges banks to invest

in expensive software and system upgrades whose cost is charged back to MTOs by raising their fees and reducing credit lines (Alani, 2017; FATF, 2015; IFAD, 2016).

De-risking has created challenges for the development of regulated and legal remittance providers. This could result in diverting flows toward informal channels, which in turn could increase AML/CFT risks. However, methods used by criminals to conceal their illegal activities and the volumes of their transactions are very different from the transactions used by immigrants to send money back home. Mixing the two could be misleading and results in adverse consequences for migrants (Nikos, 2018; World Bank, 2017).

A1.4 Technology and Remittance cost reduction

New incumbents in the MTO market that are proposing digital technologies are disrupting the sector. In 2017, these new players have nearly achieved the UN's 3% transaction fee goal.

An important element in reducing transaction costs is the use of mobile money. The GSM Association (GSMA) states that mobile money reduces the cost of international remittances by more than 50% on average compared to using global MTOs. In 2017, $200 remittances sent using a mobile money account cost an average of 2.7%, compared to 6% when using global MTOs (GSMA, 2017).

It should also be noted that the average value of international transfers sent using mobile money is relatively low ($82 in June 2015) compared to the average amount of international transfers across all channels (around $500) and thus mobile money is better suited to small remittances that can be sent and received from the convenience of a mobile phone in one's home (GSMA, 2017).

Competition, technology, UN pressure and G20 pressure to lower costs for senders has resulted in fees for transfers declining year over year since 2008.

With fees falling, MTOs are now differentiating themselves from their competitors by offering new services to increase customer loyalty (Romaldini, 2018). This diversity of services and means of transferring money, from credit card web-based transfers performed in the convenience of a remittance sender's home, to mobile money sent from a user's mobile phone connected to his bank credit card, to cryptocurrency infrastructure, is an opportunity for remittances to quickly contribute to the achievement of economic growth, financial inclusion, and women's economic empowerment for beneficiaries of the remittance flow in developing countries (WorldBank, 2019).

A1.5 Conclusion

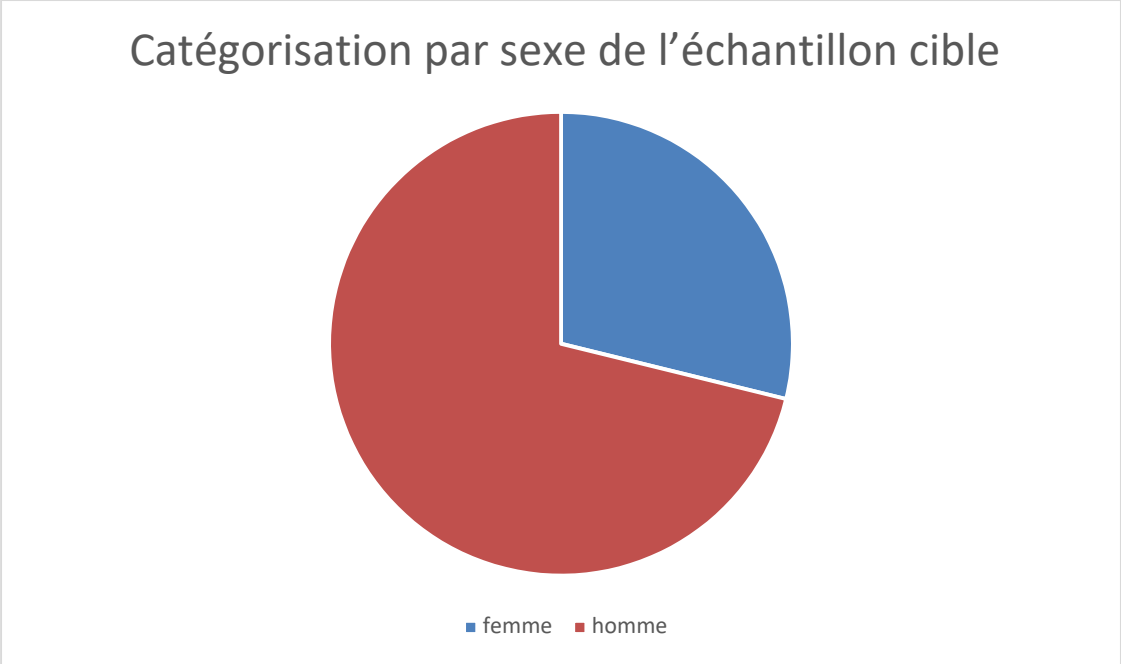
This research report showed that Africa received $37.8 billion USD in remittances in 2017 and that there is an expected 10% growth compared with 2016 (WorldBank, 2017). In Africa, remittance money inflow is now superior to foreign investment in these countries. All this money flow is having a positive impact on people lives, however the cost of sending remittance money is still high and there is an issue with the trustworthiness of the people receiving money when the remittance is sent for the accomplishment of a specific purpose. The use of new technology could sensibly reduce the cost of remitting money in Africa.

APPENDIX 2

QUANTITATIVE SURVEY RESULTS

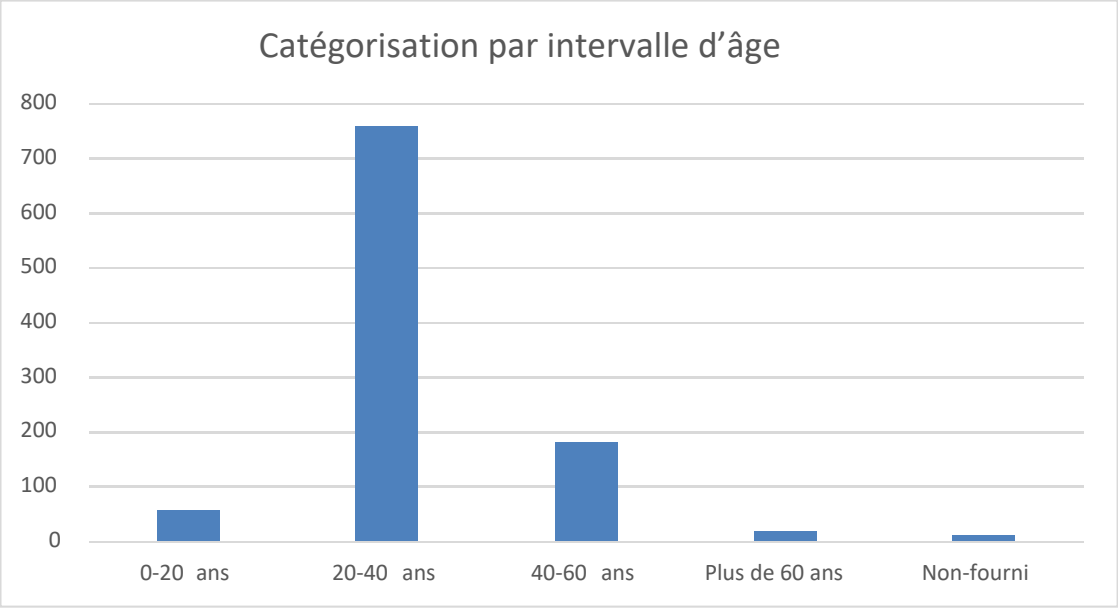
1. Catégorisation de l'échantillon par sexe de l'échantillon cible

	Nombre
Femme	297
Homme	733
Total général	1030



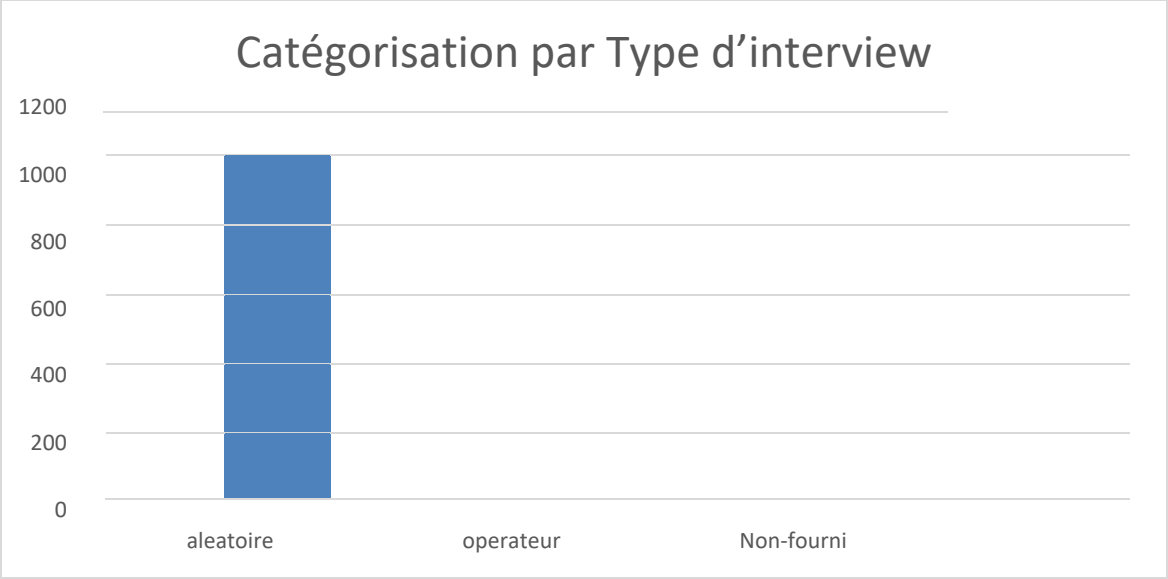
2. Catégorisation de l'échantillon par intervalle d'âge

Intervalle d'âge	Nombre
0-20 ans	59
20-40 ans	759
40-60 ans	181
Plus de 60 ans	19
Non-fourni	12
Total général	1030



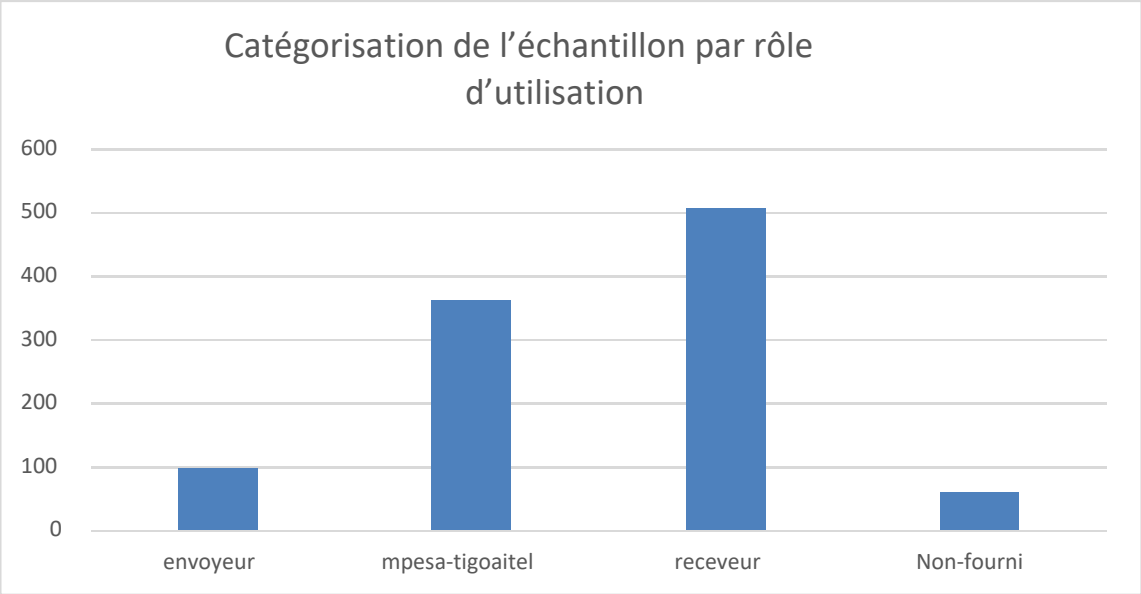
3. Catégorisation de l'échantillon par Type d'interview

Type d'inteview	Nombre
Aléatoire	1007
Operateur	22
Non-fourni	1
Total général	1030



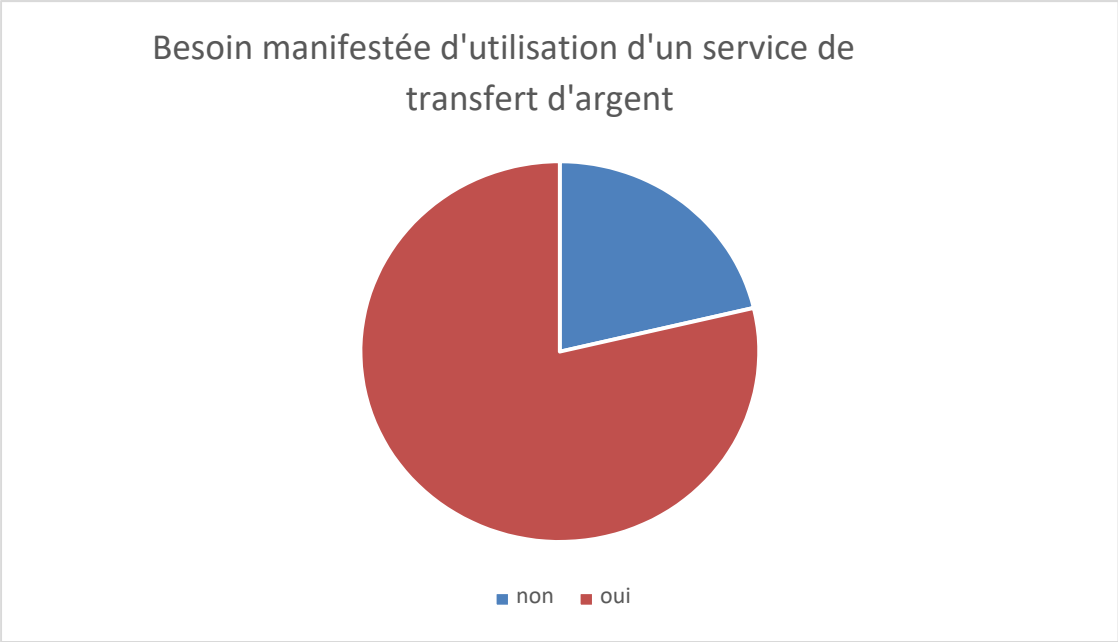
4. Catégorisation de l'échantillon par rôle d'utilisation des services de transfert d'argent

Rôle d'utilisation	Nombre
Envoyeur par MTO	99
Utilise le mobile money et non les MTO	362
Receveur par MTO	508
Non-Utilisateur	61
Total général	1030



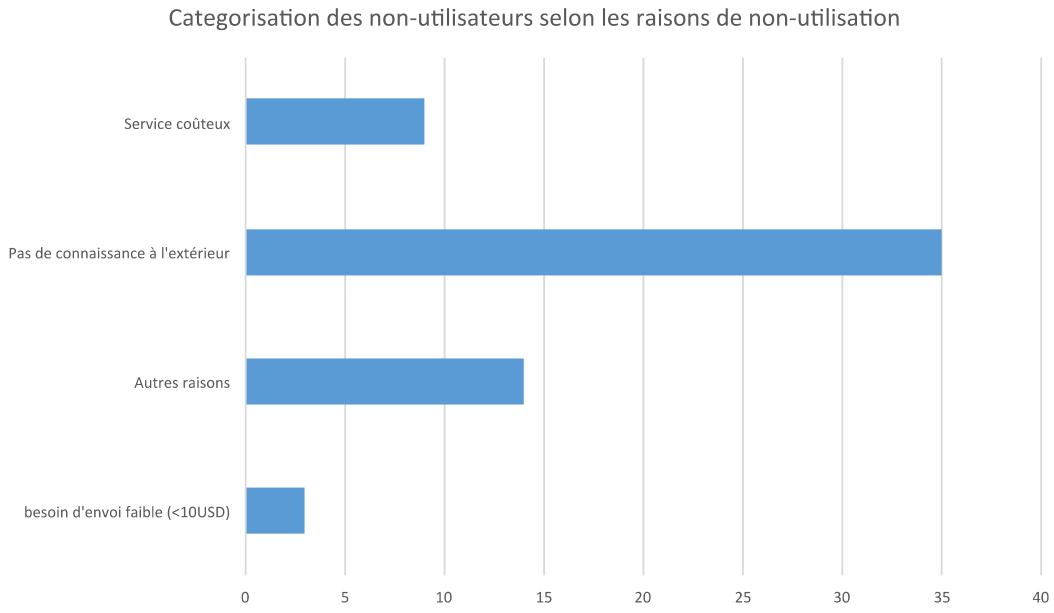
5. Catégorisation des non-utilisateurs des services de transfert d'argent selon le besoin en utilisation d'un service de transfert d'argent

Besoin manifesté	Nombre
Non je n'ai pas besoin de MTO	13
Oui j'ai besoin d'un MTO	48
Total général	61



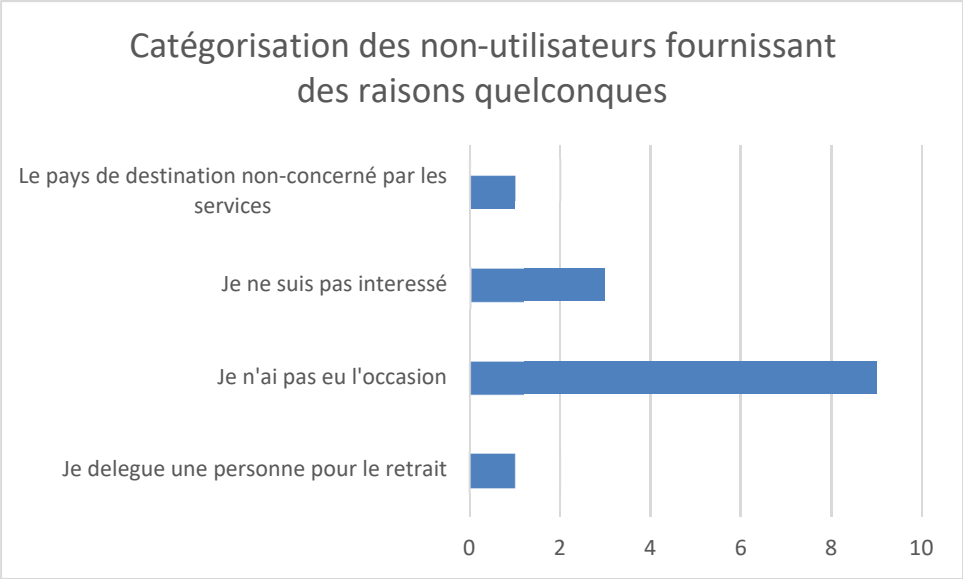
6. Catégorisation des non-utilisateurs des services de transfert d'argent selon les raisons de non-utilisation

Raison de non-utilisation	Nombre
Besoin d'envoi faible (<10USD)	3
Autres raisons	14
Pas de connaissance à l'extérieur qui pourrait m'envoyer de l'argent	35
Service coûteux	9
Total général	61



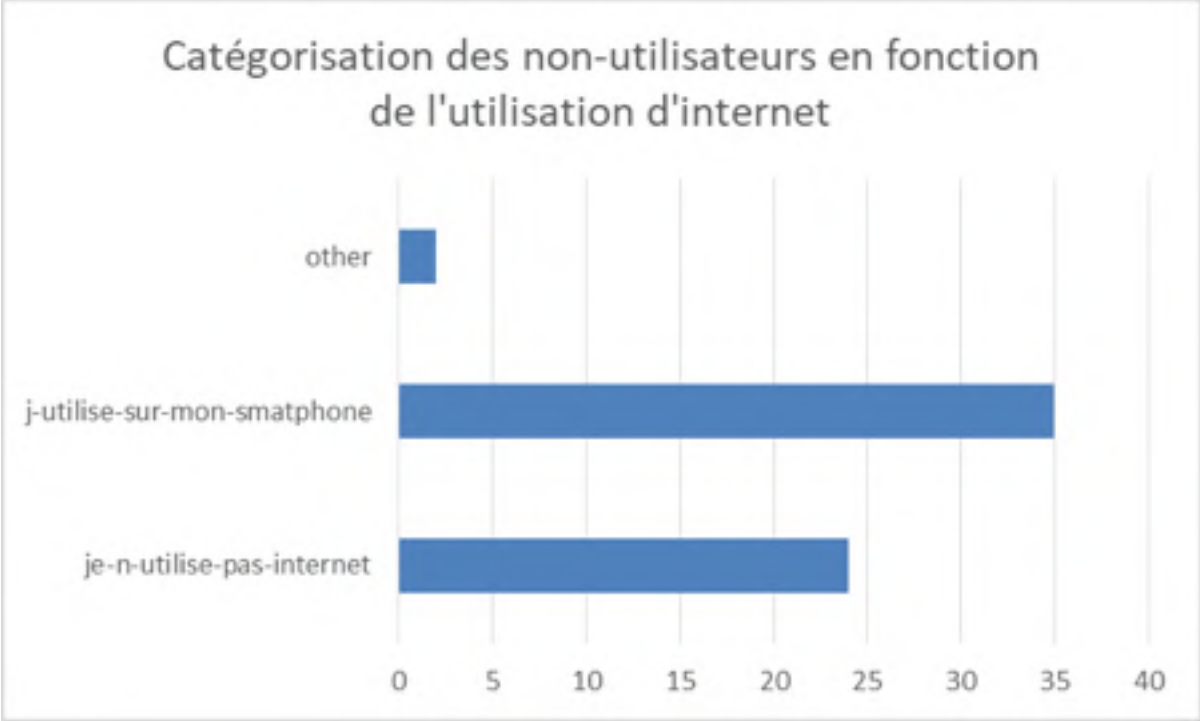
7. Catégorisation des non-utilisateurs des services de transfert d'argent fournissant des raisons quelconques

Raison quelconque des non-utilisateurs	Nombre
Je délègue une personne pour le retrait	1
Je n'ai pas eu l'occasion	9
Je ne suis pas intéressé	3
Le pays de destination non-concerné par les services	1
Total général	14



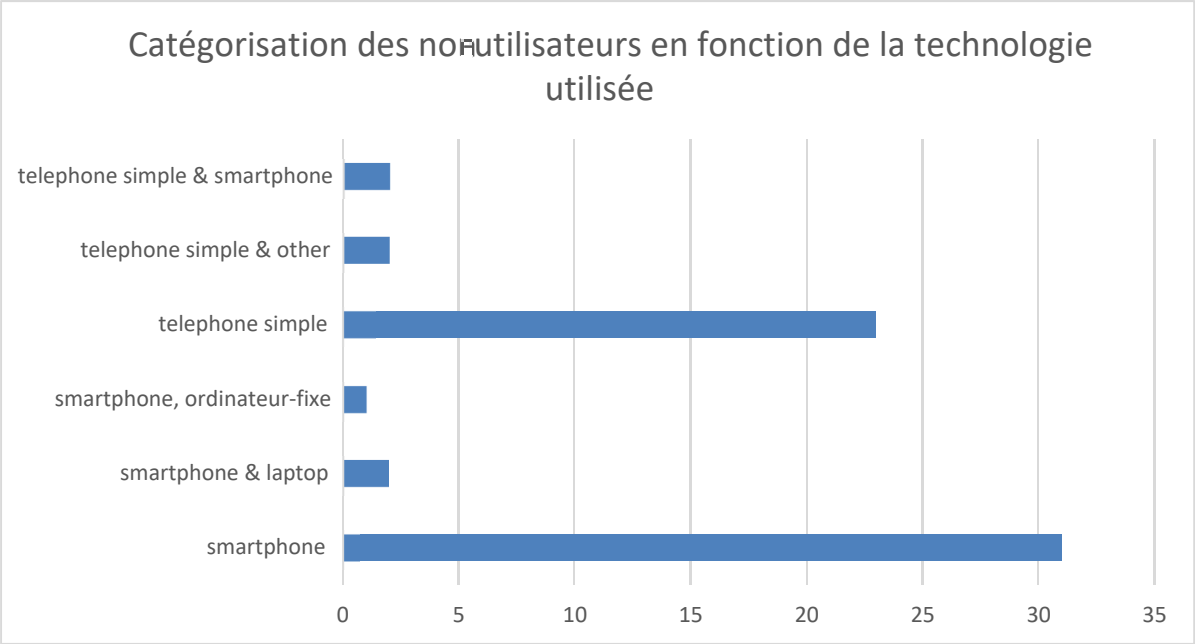
8. Catégorisation des non-utilisateurs des services de transfert d'argent en fonction de l'utilisation d'internet

Utilisation d'internet	Nombre
Je n'utilise pas internet	24
J'utilise sur mon smartphone	35
Autres	2
Total général	61



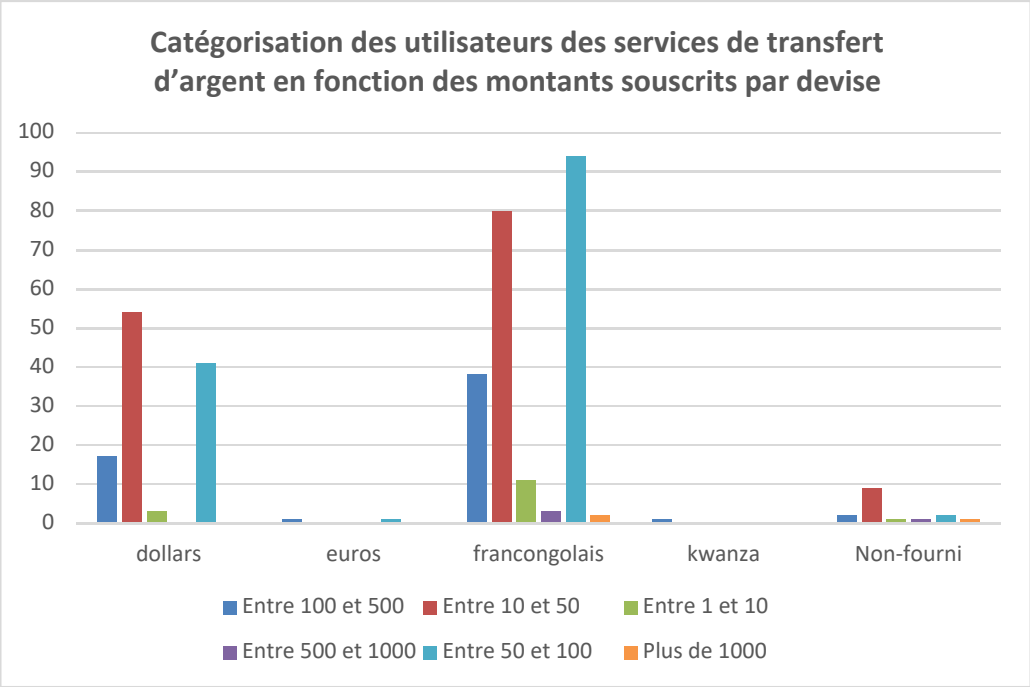
9. Catégorisation des non-utilisateurs des services de transfert d'argent en fonction de la technologie utilisée

Technologie utilisée	Nombre
smartphone	31
smartphone & laptop	2
smartphone, ordinateur-fixe	1
telephone simple	23
telephone simple & other	2
telephone simple & smartphone	2
Total général	61



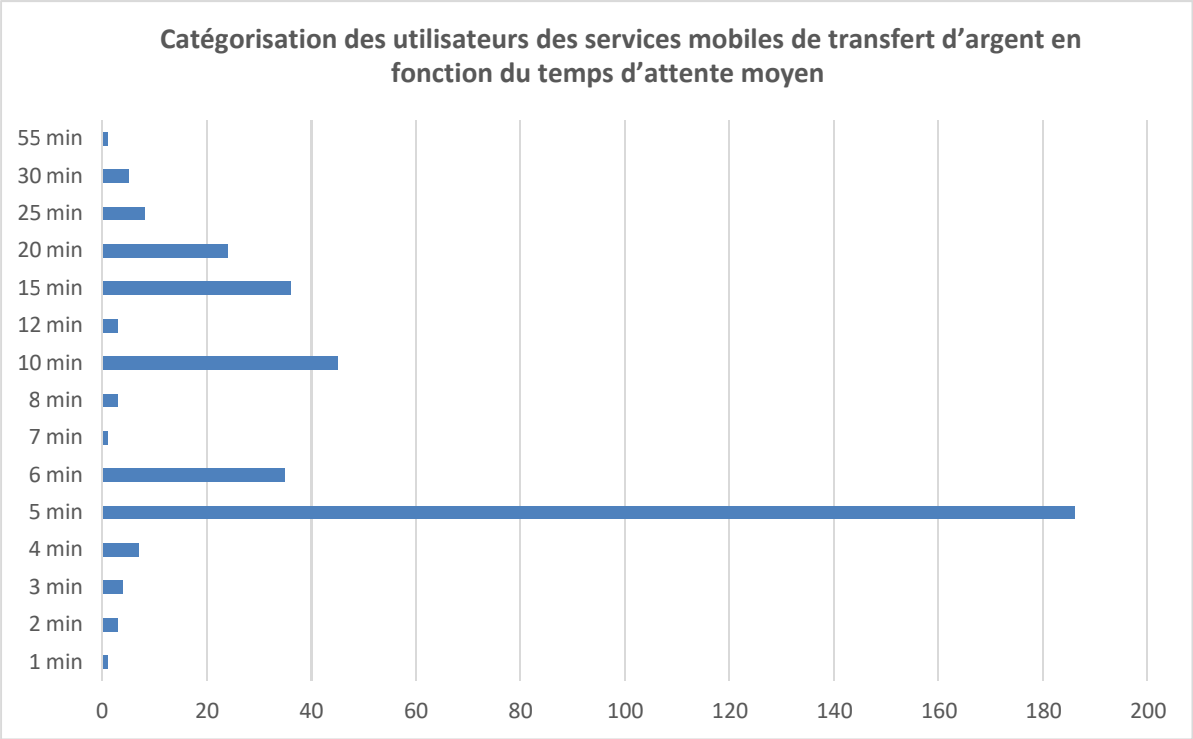
10. Catégorisation des utilisateurs des services mobiles de transfert d'argent en fonction des montants souscrits par devise

Montant	dollars	euros	fran congolais	kwanza	Non-fourni	Total général
Entre 100 et 500	17	1	38	1	2	59
Entre 10 et 50	54	0	80	0	9	143
Entre 1 et 10	3	0	11	0	1	15
Entre 500 et 1000	0	0	3	0	1	4
Entre 50 et 100	41	1	94	0	2	138
Plus de 1000	0	0	2	0	1	3
Total général	115	2	228	1	14	362



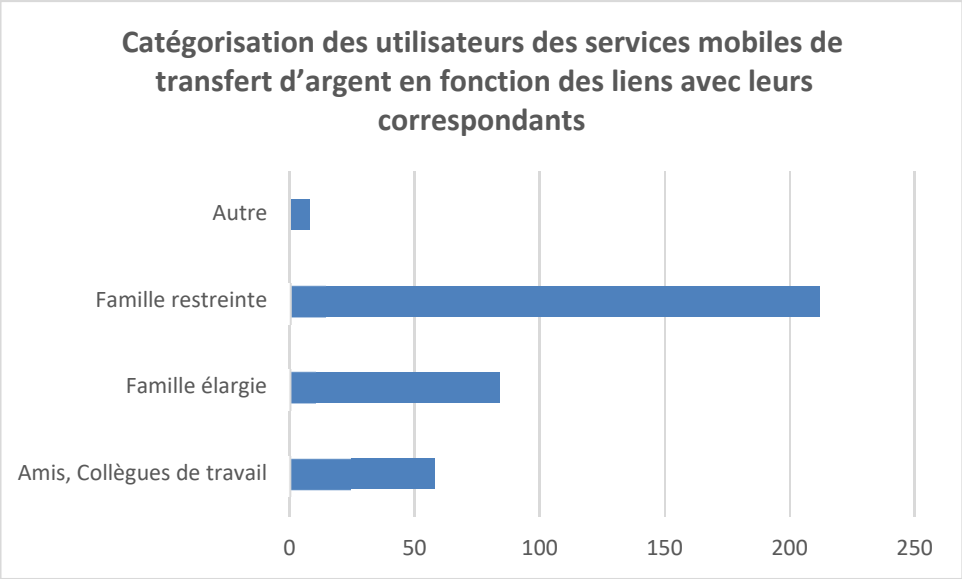
11. Catégorisation des utilisateurs des services mobiles de transfert d'argent en fonction du temps d'attente moyen

Temps d'attente moyen	Nombre
1 min	1
2 min	3
3 min	4
4 min	7
5 min	186
6 min	35
7 min	1
8 min	3
10 min	45
12 min	3
15 min	36
20 min	24
25 min	8
30 min	5
55 min	1
Total général	362



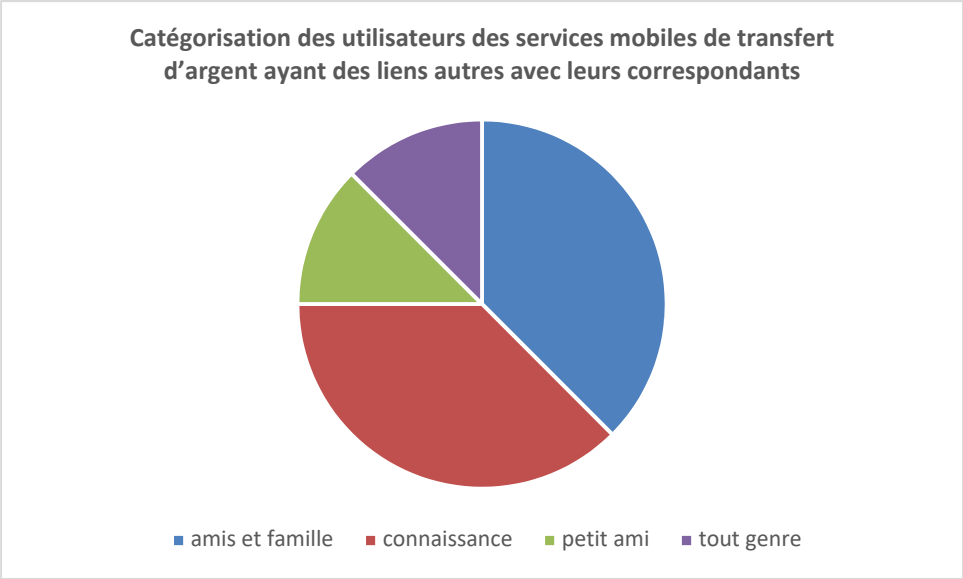
12. Catégorisation des utilisateurs des services mobiles de transfert d'argent en fonction des liens avec leurs correspondants

Lien	Nombre
Amis, Collègues de travail	58
Famille élargie	84
Famille restreinte	212
Autre	8
Total général	362



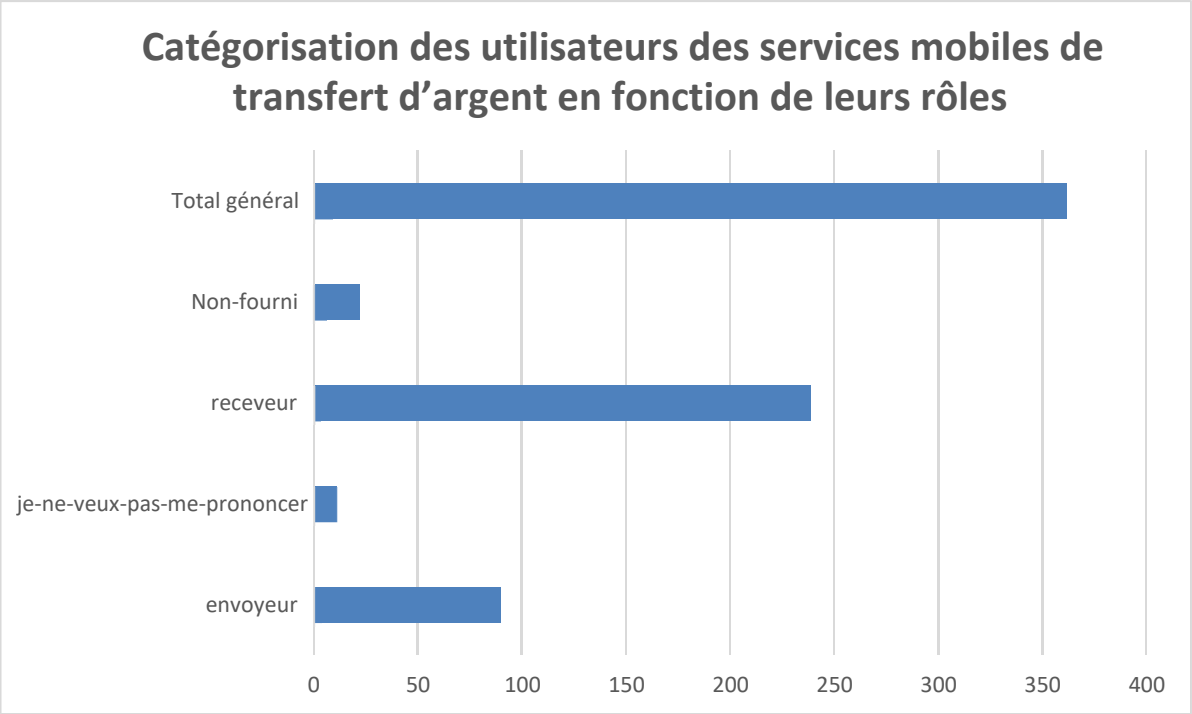
13. Catégorisation des utilisateurs des services mobiles de transfert d'argent ayant des liens autres avec leurs correspondants

Lien autre	Nombre
amis et famille	3
connaissance	3
petit ami	1
tout genre	1
Total général	8



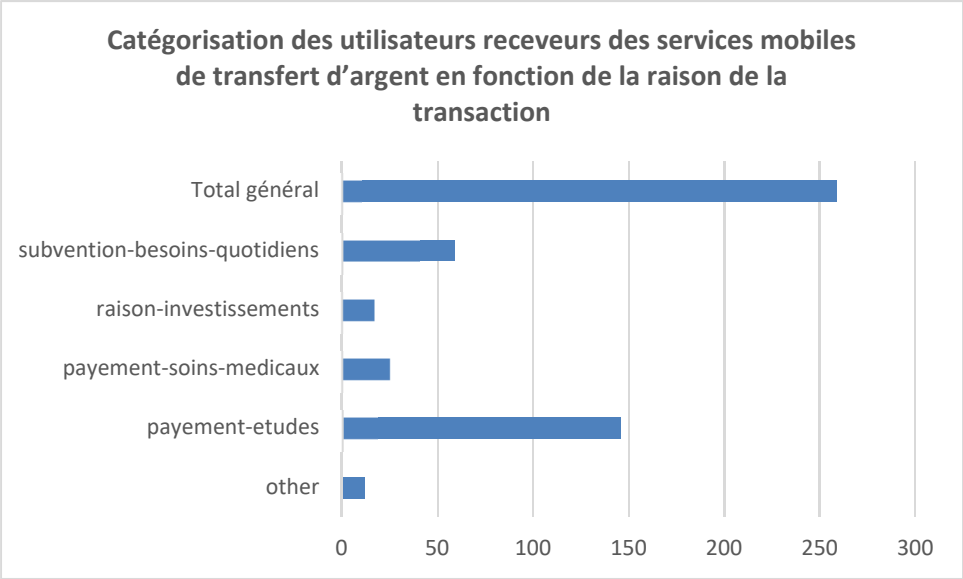
14. Catégorisation des utilisateurs des services mobiles de transfert d'argent en fonction de leurs rôles

Rôle	Nombre
envoyeur	90
je-ne-veux-pas-me-prononcer	11
receveur	239
non-fourni	22
Total général	362



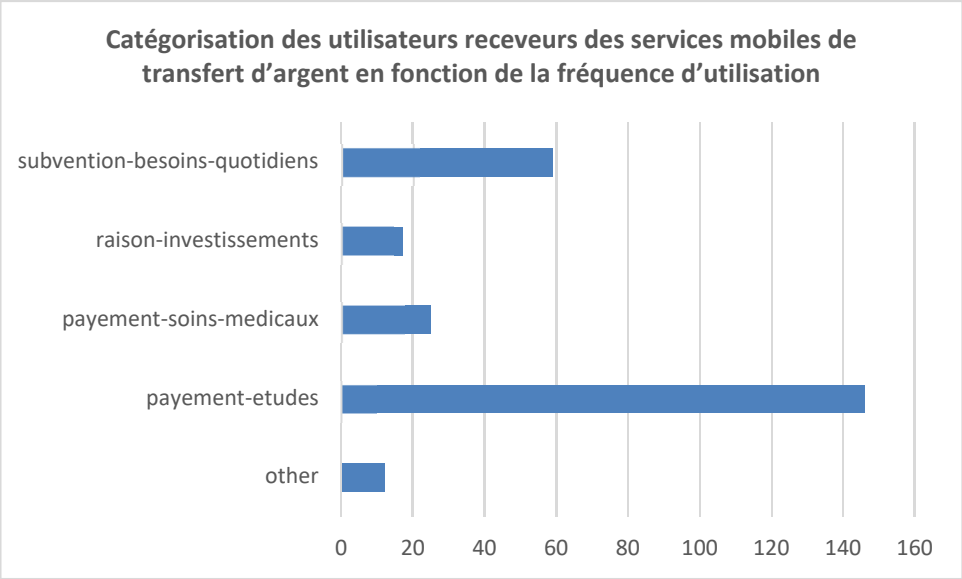
15. Catégorisation des utilisateurs receveurs des services mobiles de transfert d'argent en fonction de la raison de la transaction

Raison de la transaction	Nombre
other	12
payement-etudes	146
payement-soins-medicaux	25
raison-investissements	17
subvention-besoins-quotidiens	59
Total général	259



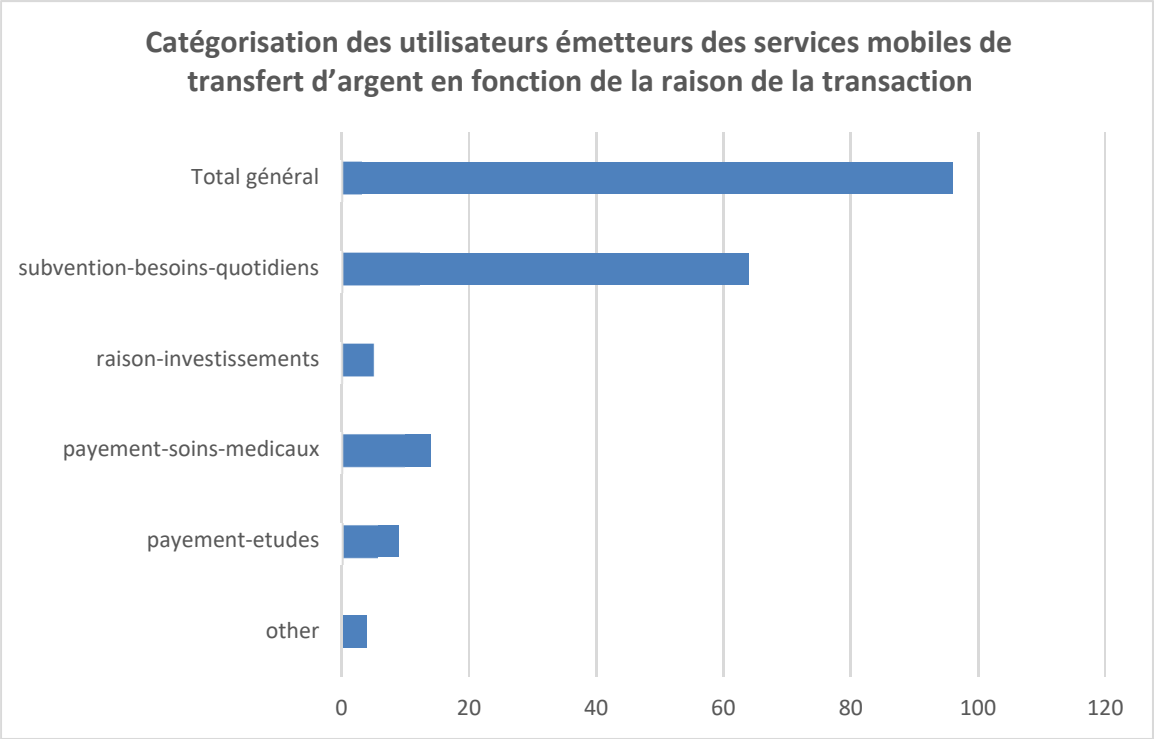
16. Catégorisation des utilisateurs receveurs des services mobiles de transfert d'argent en fonction de la fréquence d'utilisation

Fréquence d'utilisation	Nombre
other	12
payement-etudes	146
payement-soins-medicaux	25
raison-investissements	17
subvention-besoins-quotidiens	59
Total général	259



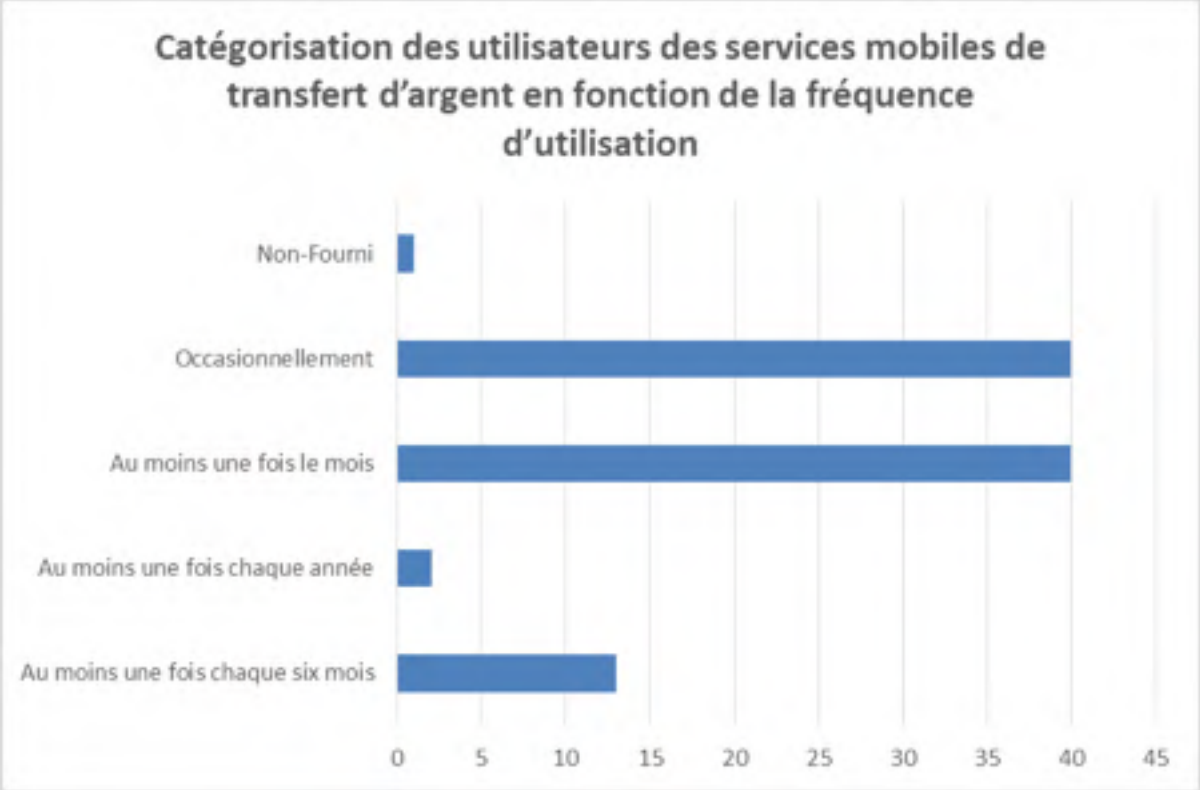
17. Catégorisation des utilisateurs émetteurs des services mobiles de transfert d'argent en fonction de la raison de la transaction

Raison de la transaction	Nombre
other	4
payement-etudes	9
payement-soins-medicaux	14
raison-investissements	5
subvention-besoins-quotidiens	64
Total général	96



18. Catégorisation des utilisateurs émetteurs des services mobiles de transfert d'argent en fonction de la fréquence d'utilisation

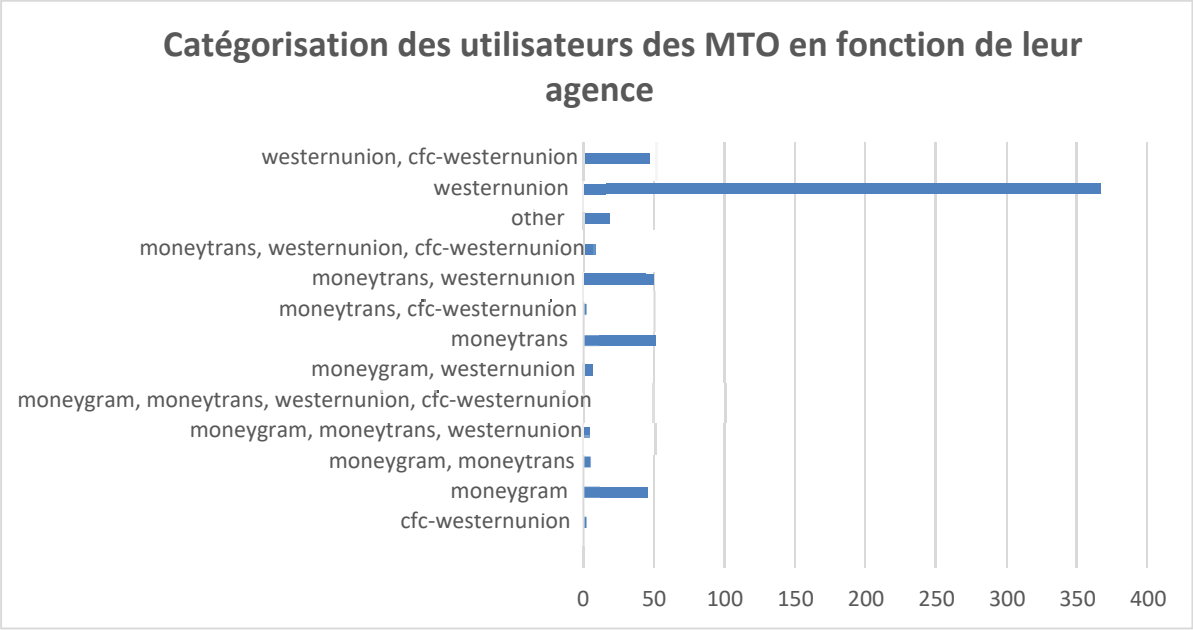
Fréquence d'utilisation	Nombre
Au moins une fois chaque six mois	13
Au moins une fois chaque année	2
Au moins une fois le mois	40
Occasionnellement	40
Non-Fourni	1
Total	96



19. Catégorisation des utilisateurs des MTO en fonction de leur agence

Agence	Nombre
cfc-westernunion	2
moneygram	46
moneygram, moneytrans	5
moneygram, moneytrans, westernunion, cfc-westernunion	4
westernunion	1
moneygram, westernunion	6
moneytrans	51
moneytrans, cfc-westernunion	2
moneytrans, westernunion	50
moneytrans, westernunion, cfc-westernunion	8

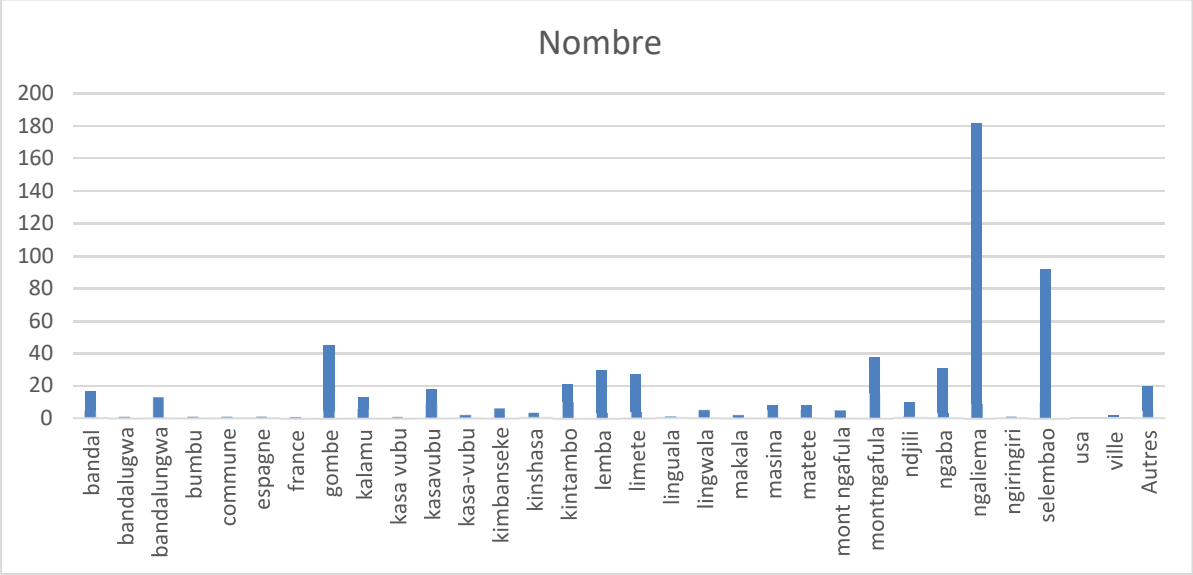
Agence	Nombre
other	19
westernunion	367
westernunion, cfc-westernunion	46
Total général	607



20. Catégorisations des utilisateurs MTO en fonction leur commune

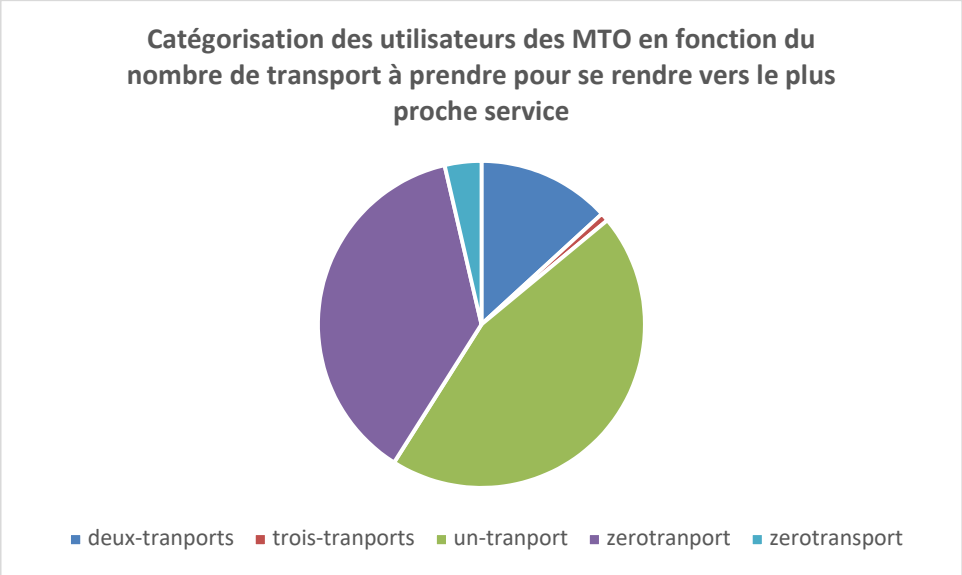
Commune	Nombre
bandal	17
bandalugwa	1
bandalungwa	13
bumbu	1
commune	1
espagne	1
france	1
gombe	45
kalamu	13
kasa vubu	1
kasavubu	18
kasa-vubu	2
kimbanseke	6
kinshasa	3
kintambo	21
lemba	30
limete	27
linguala	1
lingwala	5
makala	2
masina	8
matete	8
mont ngafula	5
montngafula	38

Commune	Nombre
ndjili	10
ngaba	31
ngaliema	182
ngiringiri	1
selembao	92
usa	1
ville	2
Autres	20
Total général	607



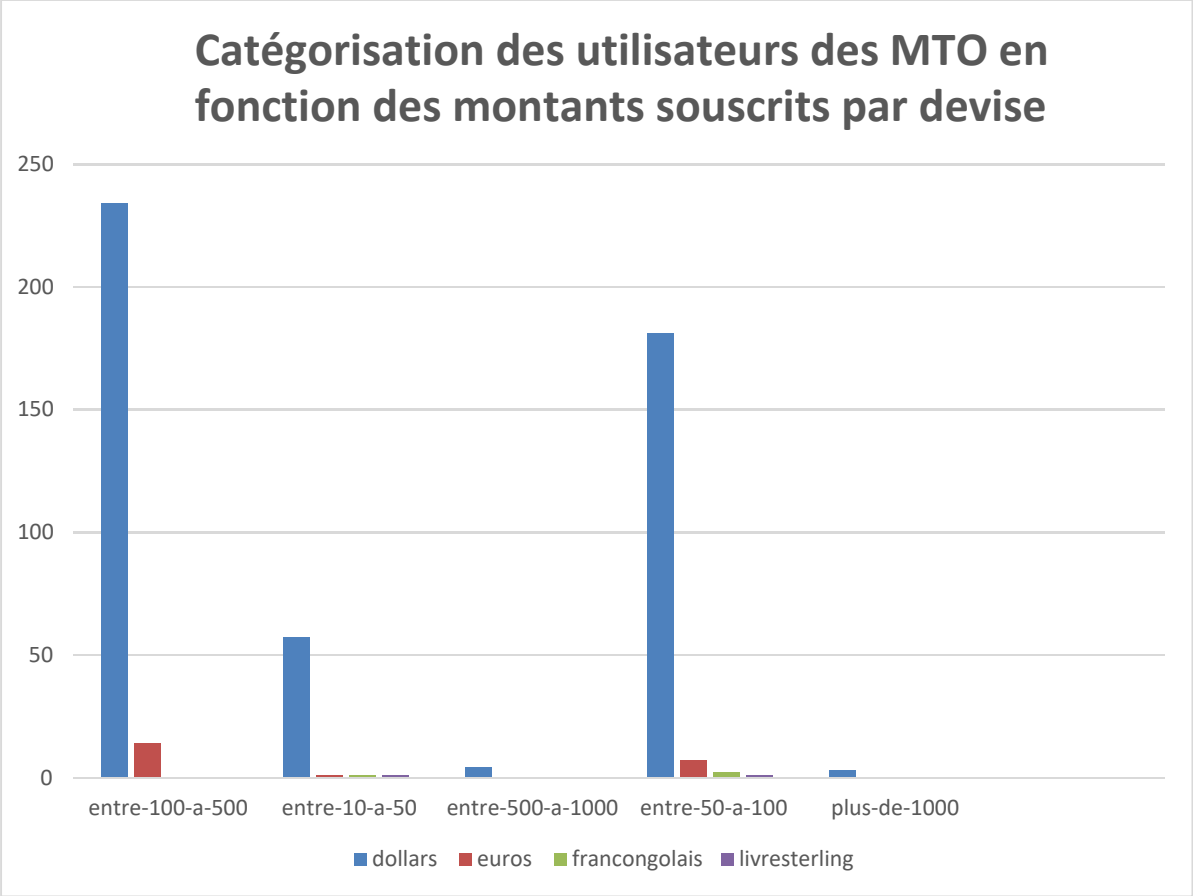
21. Catégorisation des utilisateurs des MTO en fonction du nombre de transport à prendre pour se rendre vers le plus proche service

Nombre de transport	Nombre
deux-tranports	80
trois-tranports	5
un-tranport	273
zerotranport	227
zerotransport	22
Total général	607



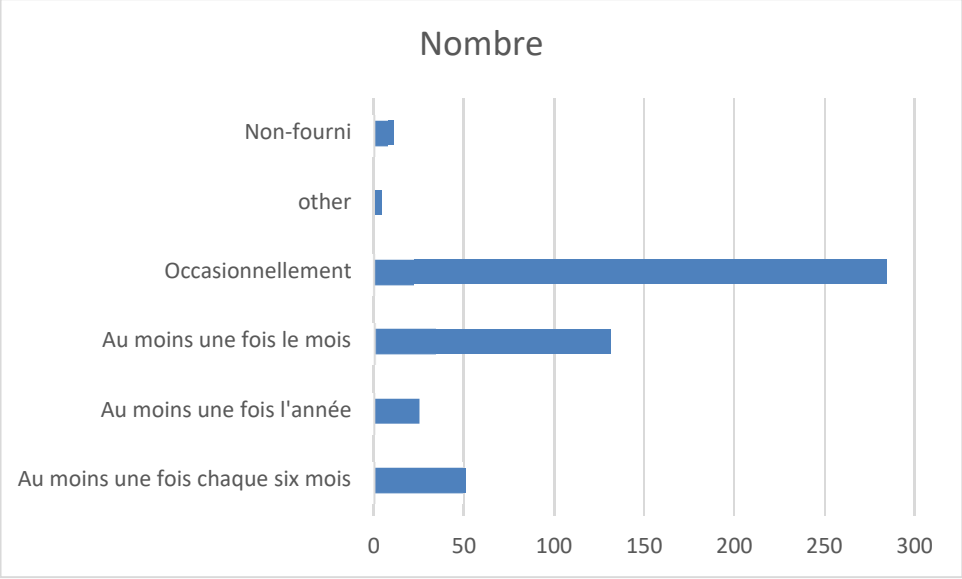
22. Catégorisation des utilisateurs des MTO en fonction des montants souscrits par devise pour la réception des fonds

Montant	dollars	euros	fran congolais	livre sterling	Total général
entre-100-a-500	234	14			248
entre-10-a-50	57	1	1	1	60
entre-500-a-1000	4				4
entre-50-a-100	181	7	2	1	191
plus-de-1000	3				3
non-Fourni					2
Total général	479	22	3	2	508



23. Catégorisation des utilisateurs des MTO en fonction de la fréquence de réception de fonds

Fréquence	Nombre
Au moins une fois chaque six mois	51
Au moins une fois l'année	25
Au moins une fois le mois	131
Occasionnellement	284
other	4
Non-fourni	13
Total général	508

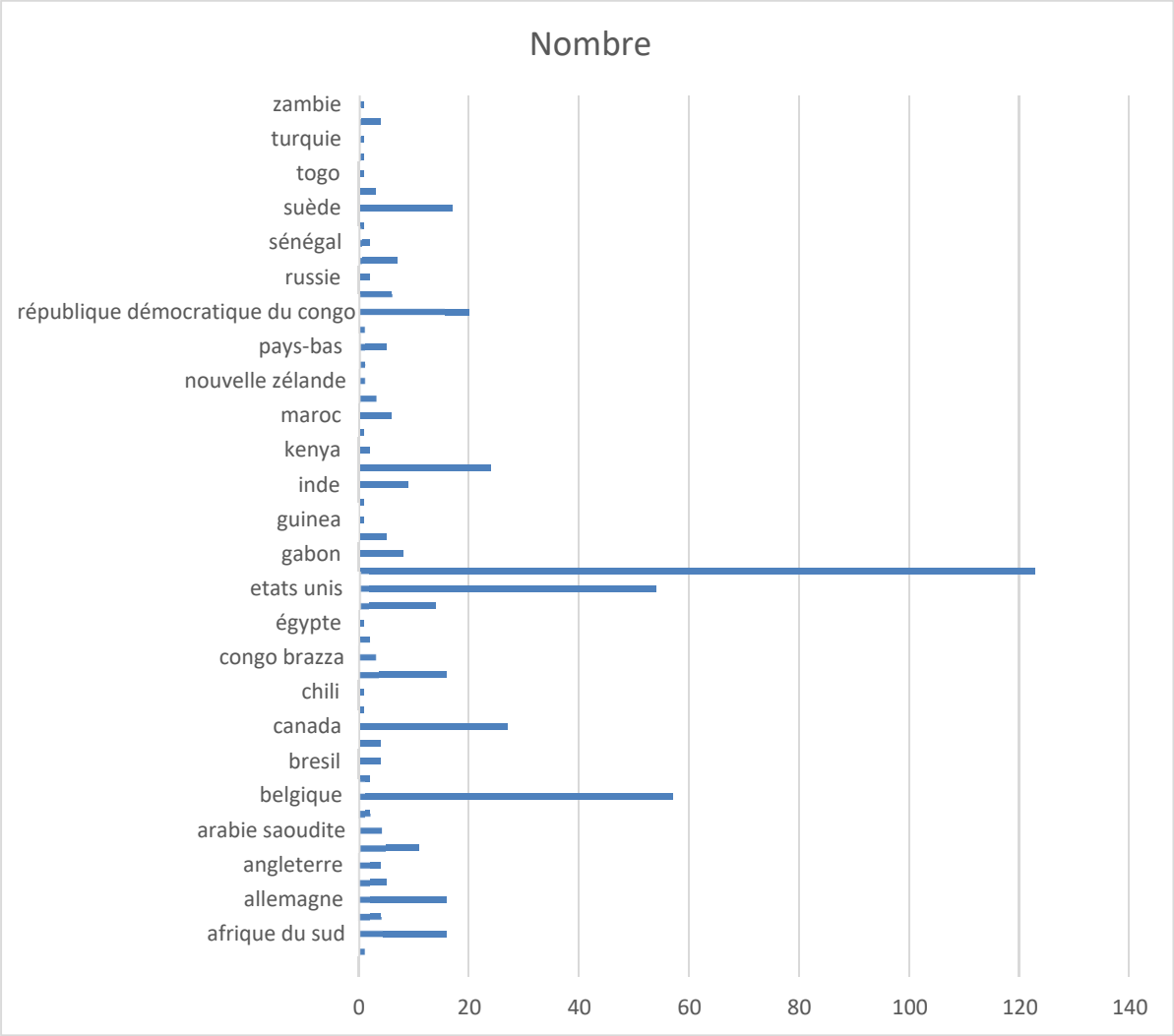


24. Catégorisation des utilisateurs des MTO en fonction du pays de réception

Pays	Nombre
non-fourni	2
afrique du nord	1
afrique du sud	16
algérie	4
allemagne	16
allemand	5
angleterre	4
angola	11
arabie saoudite	4
autriche	2
belgique	57
benin	2
bresil	4
cameroun	4
canada	27
cap vert	1
chili	1
chine	16
congo brazza	3
corée	1
côte d'ivoire	2
égypte	1
espagne	14
etats unis	54
france	123

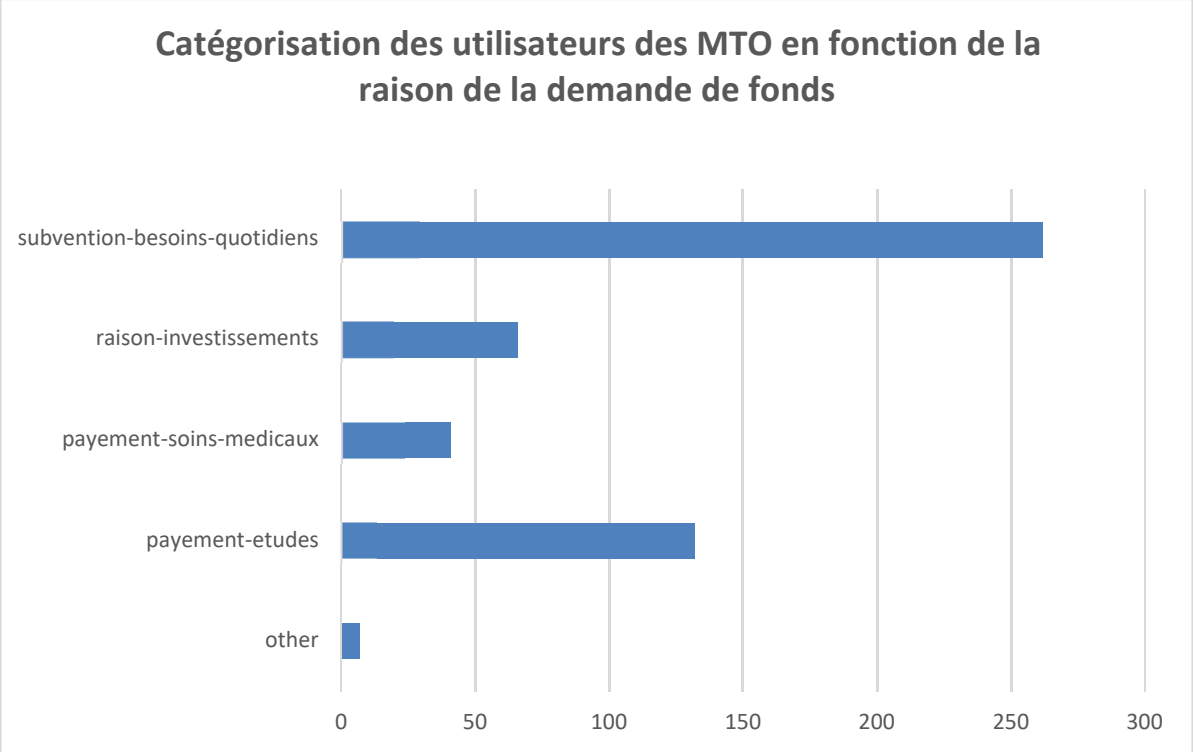
Pays	Nombre
gabon	8
grèce	5
guinea	1
île maurice	1
inde	9
italie	24
kenya	2
maroc	6
mexique	3
nouvelle zélande	1
ouganda	1
pays-bas	5
portugal	1
république démocratique du congo	20
royaume-uni	6
russie	2
rwanda	7
sénégal	2
singapour	1
suède	17
suisse	3
togo	1
tunisie	1
turquie	1
usa	4
zambie	1

Pays	Nombre
Total général	508



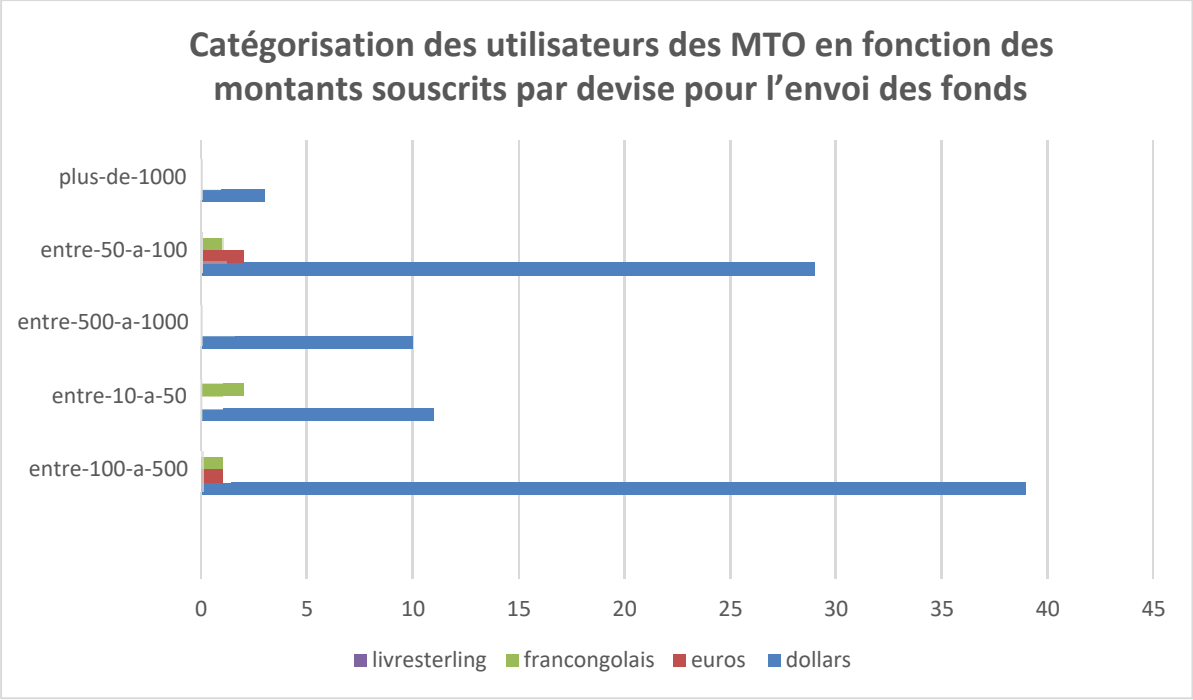
25. Catégorisation des utilisateurs des MTO en fonction de la raison de la demande de fonds

Raison de la demande	Nombre
other	7
payement-etudes	132
payement-soins-medicaux	41
raison-investissements	66
subvention-besoinsquotidiens	262
Total général	508



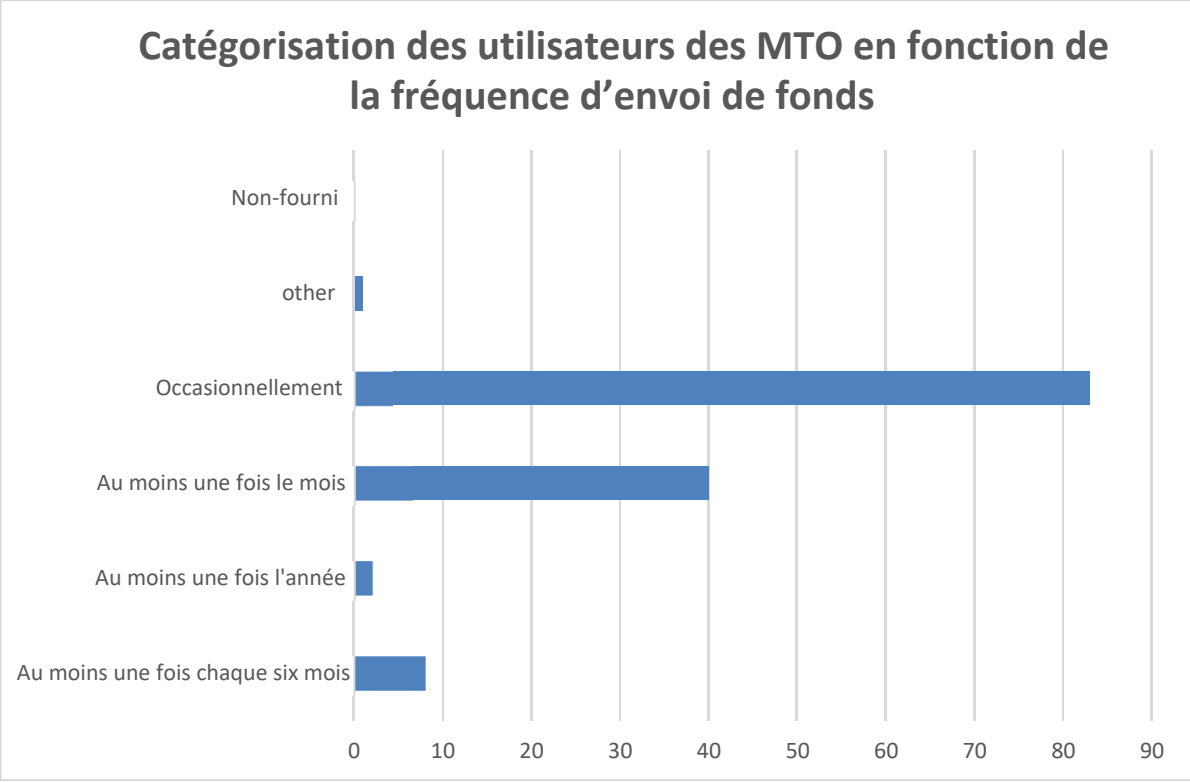
26. Catégorisation des utilisateurs des MTO en fonction des montants souscrits par devise pour l'envoi des fonds

Montant	dollars	euros	fran congolais	livre sterling	Total général
entre-100-a-500	39	1	1	0	41
entre-10-a-50	11	0	2	0	13
entre-500-a-1000	10	0	0	0	10
entre-50-a-100	29	2	1	0	32
plus-de-1000	3	0	0	0	3
non-fourni					35
Total général	92	3	4	0	99



27. Catégorisation des utilisateurs des MTO en fonction de la fréquence d'envoi de fonds

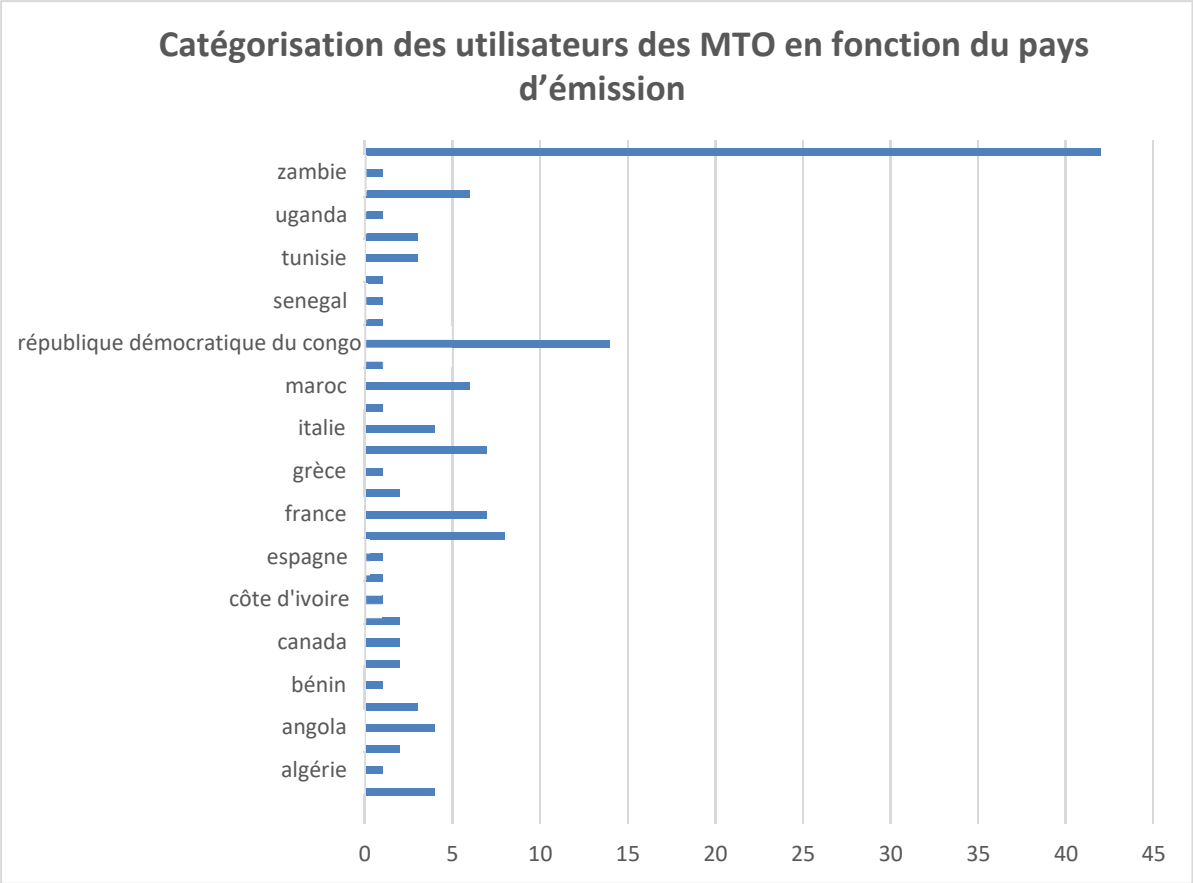
Fréquence	Nombre
Au moins une fois chaque six mois	8
Au moins une fois l'année	2
Au moins une fois le mois	40
Occasionnellement	83
Other	1
Non-fourni	0
Total général	134



28. Catégorisation des utilisateurs des MTO en fonction du pays d'émission

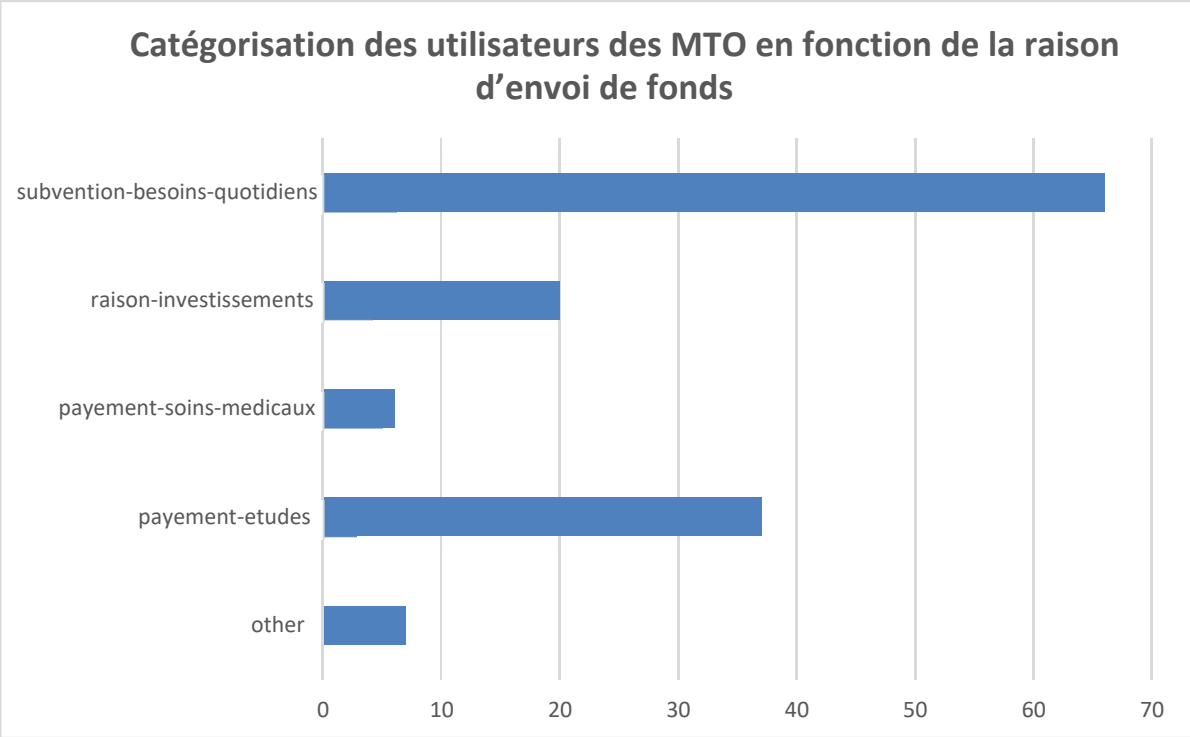
Pays	Nombre
afrique du sud	4
algérie	1
allemagne	2
angola	4
belgique	3
bénin	1
cameroun	2
canada	2
chine	2
côte d'ivoire	1
dubai	1
espagne	1
états-unis	8
france	7
gabon	2
grèce	1
inde	7
italie	4
kenya	1
maroc	6

nouvelle zélande	1
république démocratique du congo	14
russia	1
senegal	1
suisse	1
tunisie	3
turquie	3
uganda	1
usa	6
zambie	1
non-fourni	42
Total général	134



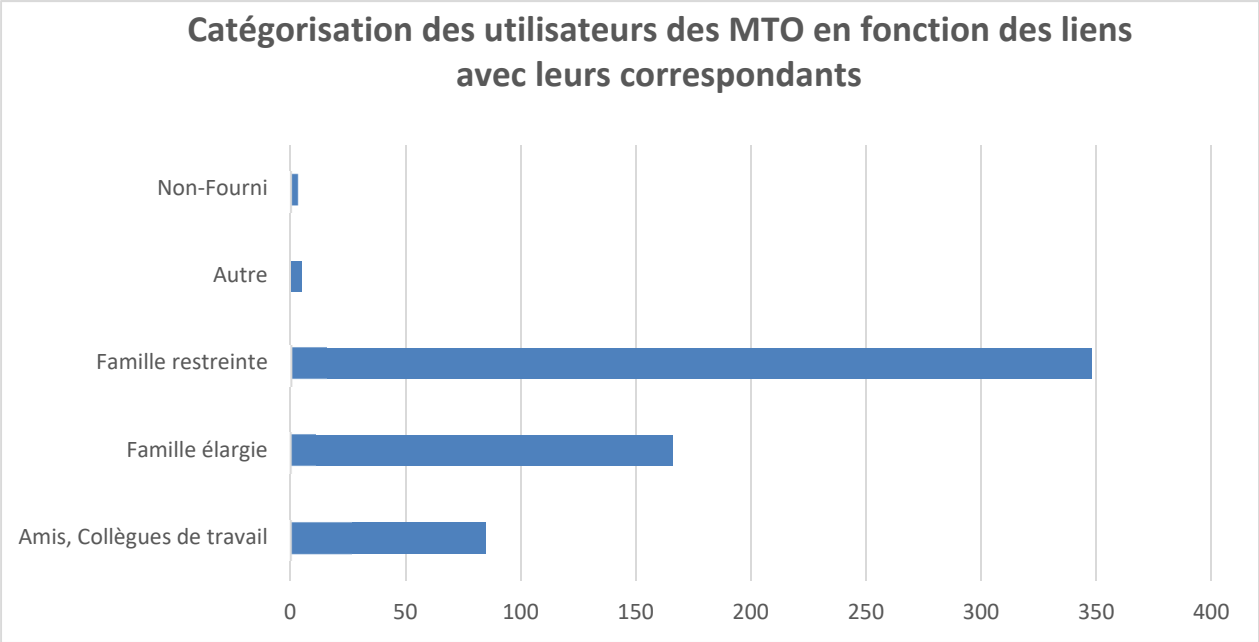
29. Catégorisation des utilisateurs des MTO en fonction de la raison d'envoi de fonds

Raison	Nombre
other	7
payement-etudes	37
payement-soins-medicaux	6
raison-investissements	20
subvention-besoinsquotidiens	66
Total général	134



30. Catégorisation des utilisateurs des MTO en fonction des liens avec leurs correspondants

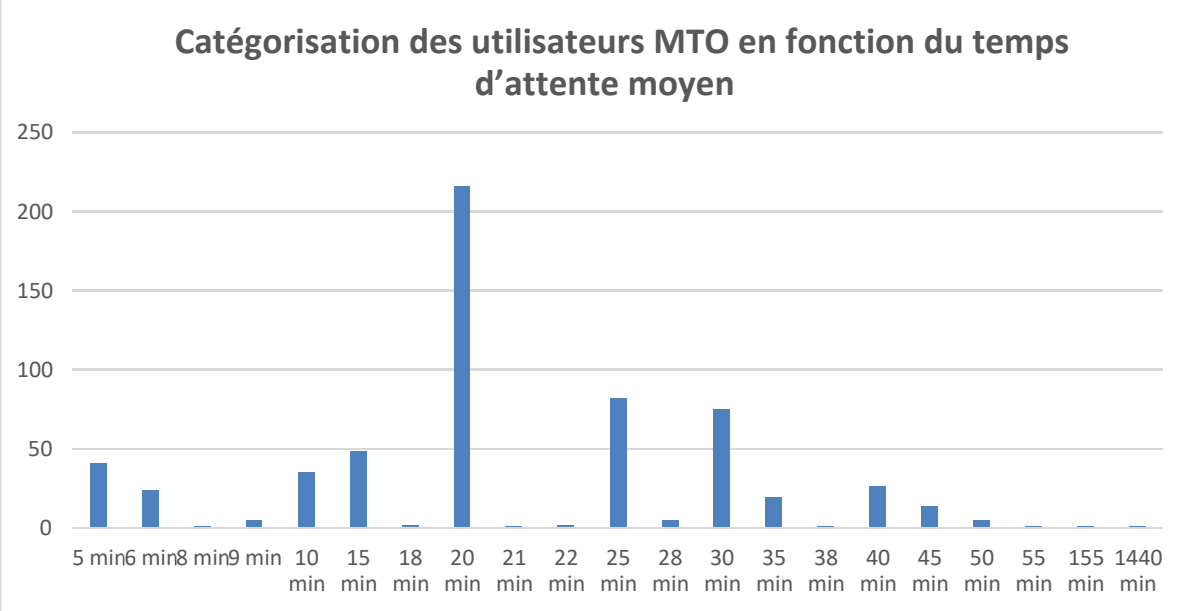
Lien	Nombre
Amis, Collègues de travail	85
Famille élargie	166
Famille restreinte	348
Autre	5
Non-Fourni	3
Total général	607



31. Catégorisation des utilisateurs MTO en fonction du temps d'attente moyen

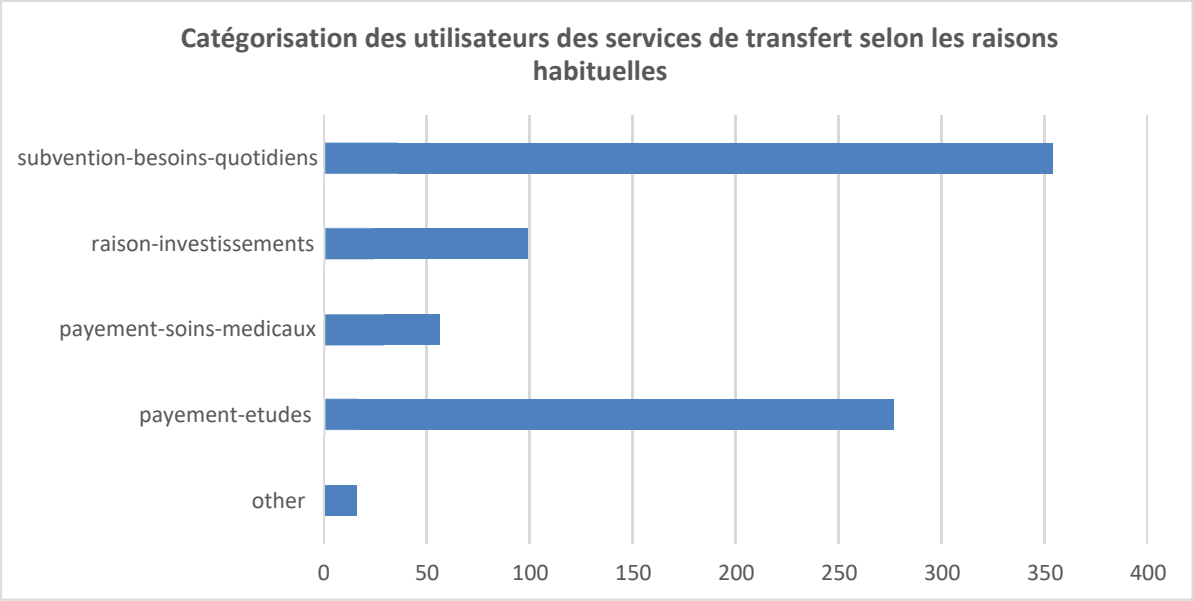
Temps d'attente moyen	Nombre
5 min	41
6 min	24
8 min	1
9 min	5
10 min	35
15 min	48
18 min	2
20 min	216
21 min	1
22 min	2
25 min	82
28 min	5
30 min	75

Temps d'attente moyen	Nombre
35 min	19
38 min	1
40 min	26
45 min	14
50 min	5
55 min	1
155 min	1
1440 min	1
Non-Fourni	2
Total général	607



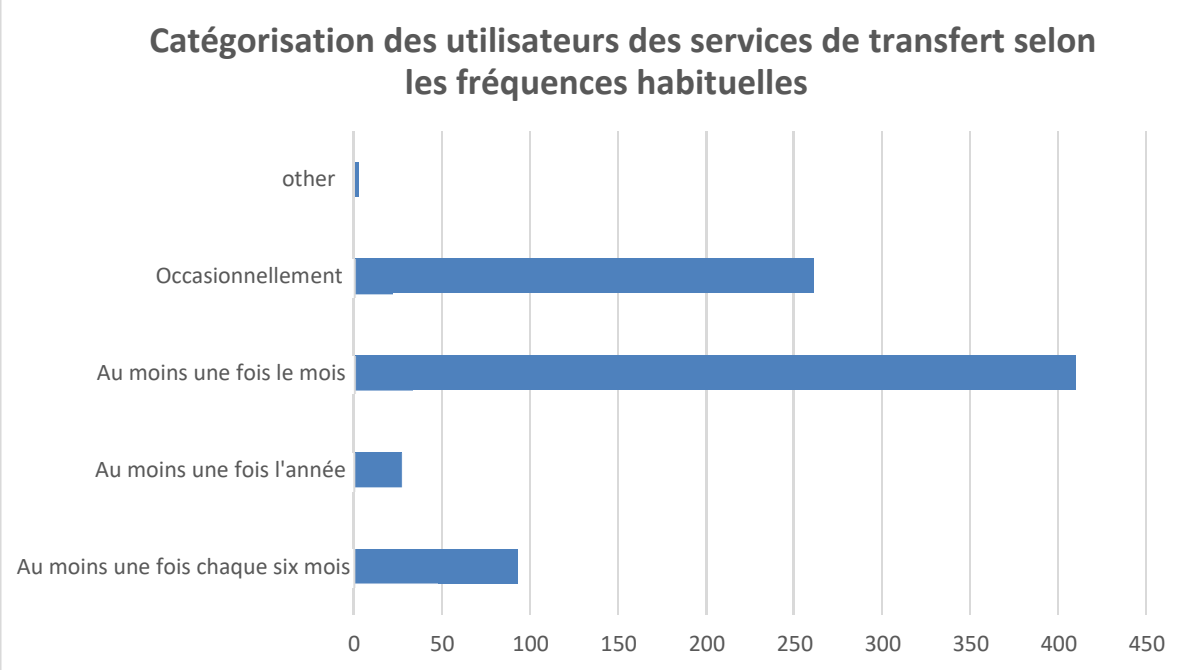
32. Catégorisation des utilisateurs des services de transfert selon les raisons habituelles

Raison	Nombre
other	16
payement-etudes	277
payement-soins-medicaux	56
raison-investissements	99
subvention-besoinsquotidiens	354
Non-Fourni	167
Total général	969



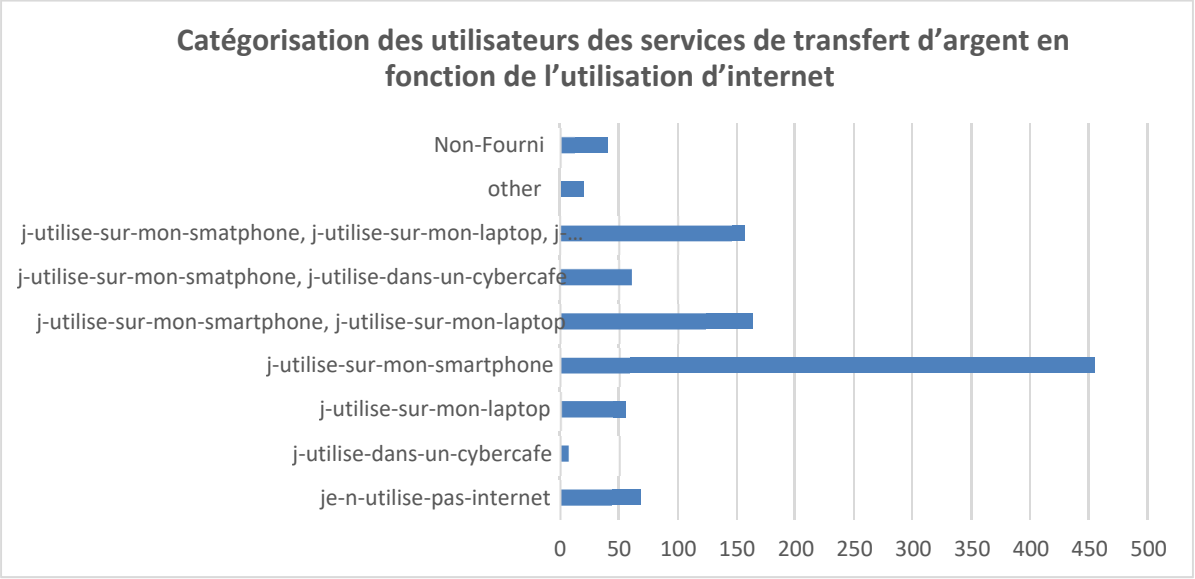
33. Catégorisation des utilisateurs des services de transfert selon les fréquences habituelles

Fréquence	Nombre
Au moins une fois chaque six mois	93
Au moins une fois l'année	27
Au moins une fois le mois	410
Occasionnellement	261
other	3
Non-fourni	175
Total général	969



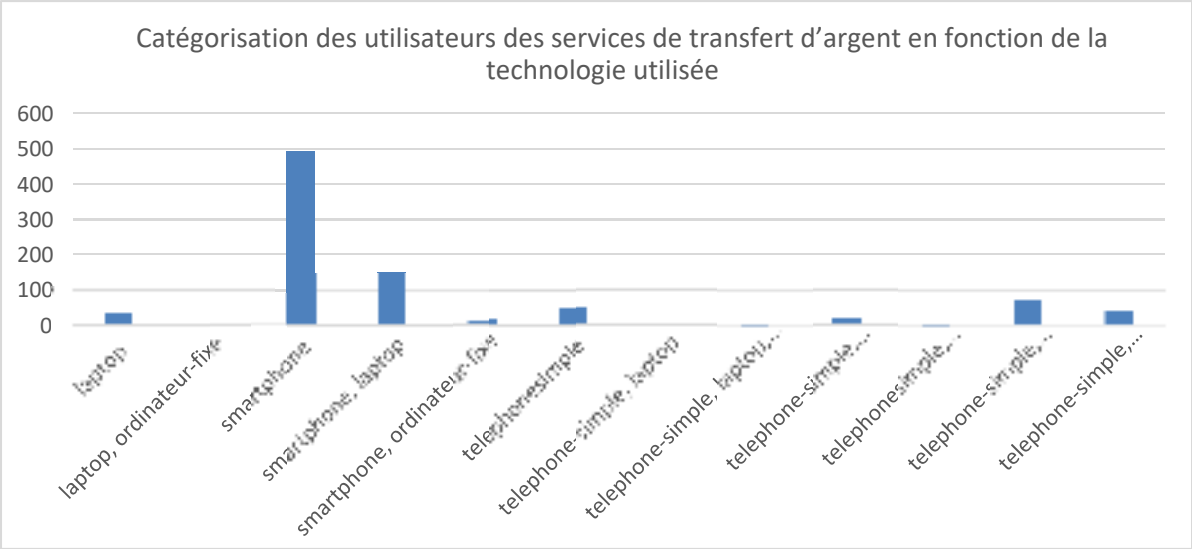
34. Catégorisation des utilisateurs des services de transfert d'argent en fonction de l'utilisation d'internet

Utilisation d'internet	Nombre
je-n-utilise-pas-internet	69
j-utilise-dans-un-cybercafe	7
j-utilise-sur-mon-laptop	56
j-utilise-sur-mon-smartphone	455
j-utilise-sur-mon-smartphone, j-utilise-sur-mon-laptop	164
j-utilise-sur-mon-smatphone, j-utilise-dans-un-cybercafe	61
j-utilise-sur-mon-smatphone, j-utilise-sur-mon-laptop, j-utilise-dans-un- cybercafe	157
other	20
Non-Fourni	41
Total général	1030



35. Catégorisation des utilisateurs des services de transfert d'argent en fonction de la technologie utilisée

Étiquettes de lignes	Nombre
Non-Fourni	151
laptop	32
laptop, ordinateur-fixe	4
smartphone	492
smartphone, laptop	149
smartphone, ordinateur-fixe	17
telephonesimple	52
telephone-simple, laptop	5
telephone-simple, laptop, ordinateur-fixe	1
telephone-simple, ordinateur-fixe	17
telephonesimple, smartphone	1
telephone-simple, smartphone, laptop	72
telephone-simple, smartphone, laptop, ordinateur-fixe	37
Total Général	1030


Catégorisation des utilisateurs des services de transfert d'argent en fonction de la technologie utilisée
600
500
400
300
200
100
0
laptop
laptop, ordinateur-fixe
smartphone
smartphone, laptop
smartphone, ordinateur-fixe
telephonesimple
telephone-simple, laptop
telephone-simple, laptou...
telephone-simple,...
telephonesimple,...
telephone-simple,...
telephone-simple,...

APPENDIX 3

LIST OF SMART CONTRACT VULNERABILITY TEST SCENARIOS

	Exploit	Type	Function/Link
1	initial contract deploy on blockchain	functionality	contract_init()
2	authentification and connection to wallet using email address and password	functionality	wallet_retrieve_info()
3	getting transaction list on deploy contract on blockchain	functionality	wallet_contract_list()
4	getting account list of beneficiary from blockchain	functionality	select_beneficiary_wallet()
5	saving amount and code message before posting to the deploy contract	functionality	transaction_operation_perform()
6	retrieving transaction received from blockchain	functionality	wallet_retrieve_transaction()
7	signing transaction(beneficiary)	functionality	transaction_operation_signing()
8	checking code message while signing transaction	functionality	transaction_operation_signing()
9	retrieving status of transaction	functionality	transaction_operation_checking()
10	ordering transaction to the service provider	functionality	transaction_operation_ordering()
11	receiving transaction from the contract	functionality	transaction_operation_ordering()
12	receiving transaction from the contract signing transaction (service provider)	functionality	transaction_operation_signing()

	Exploit	Type	Function/Link
13	getting account list of service provider from blockchain	functionality	select_beneficiary_wallet()
14	unprotected functions	functionality	unchecked_suicide, eher_send
15	missing check on CALL return value	functionality	unchecked_retval
16	checking untrusted transaction	functionality	external calls to untrusted contracts
17	multiple sends in a single transaction	functionality	external calls to untrusted contracts
18	external call to untrusted contract	functionality	external calls to untrusted contracts
19	Delegatecall or callcode to untrusted contract	functionality	external calls to untrusted contracts
20	integer overflow/underflow	functionality	validate arithmetic
21	timestamp dependence	Vulnerability	dependence on predictable variable
22	Re-Entrancy	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5148
23	Arithmetic Over/Under Flows	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5149
24	Unexpected Ether	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5150
25	Delegatecall	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5151

	Exploit	Type	Function/Link
26	Default Visibilities	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5152
27	Entropy Illusion	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5153
28	External Contract Referencing	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5154
29	Short Address/Parameter Attack	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5155
30	Unchecked CALL Return Values	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5156
31	Race Conditions/Front Running	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5157
32	Denial Of Service (DOS)	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5158
33	Block Timestamp Manipulation	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5159
34	Constructors with Care	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5160

	Exploit	Type	Function/Link
35	Uninitialized Storage Pointers	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilities- their-fixes-and-real-worldexamples-f3210eba5161
36	Floating Points and Precision	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5162
37	Tx.Origin Authentification	Vulnerability	https://hackernoon.com/hackpedia16-solidity-hacks-vulnerabilitiestheir-fixes-and-real-worldexamples-f3210eba5163
38	Frozen ether	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
39	Upgradable contract	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
40	DoS with unexpected revert	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
41	Integer overflow and underflow	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
42	Manipulated balance	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
43	Erroneous visibility	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
44	Unprotected suicide	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
45	Leaking Ether to arbitrary address	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
46	Secrecy failure	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
47	Insufficient signature information	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
48	DoS with unbounded operations	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
49	Erroneous constructor name	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf

	Exploit	Type	Function/Link
50	Type casts	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
51	Outdated compiler version	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
52	Ether lost to orphan address	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
53	Call-stack depth limit	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
54	Under-priced opcodes	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
55	Transaction ordering dependence	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
56	Timestamp dependence	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
57	Generating randomness	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
58	Indistinguishable chains	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
59	Empty account in the state trie	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
60	Outsourceable puzzle	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
61	51% hashrate	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
62	Fixed consensus termination	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
63	DoS with bloc Stuffing	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
64	Rewards for uncle blocks	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
65	Unlimited nodes creation	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
66	Uncapped incoming connections	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
67	Public peer selection	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf

	Exploit	Type	Function/Link
68	Sole block synchronization	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
69	RPC API exposure	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
70	Weak password	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
71	Cross-Site Scripting	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
72	Unvalidated URL redirection	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
73	Broken access control	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
74	Unreliable Border Gateway Protocol messages	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
75	Sensitive Domain Name System	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
76	Smart contract programming	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
77	Solidity language and tool chain	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
78	Ethereum design and implementation	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
79	Human, usability and networking factors	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf
80	Cross-cutting analysis	Vulnerability	https://arxiv.org/pdf/1908.04507.pdf

LIST OF BIBLIOGRAPHICAL REFERENCES

Acdx. (2008). Digital Signature Diagram. Available at: https://commons.wikimedia.org/wiki/File:Digital_Signature_diagram.svg

Alani, A. (2017). De-Risking 'Phenomenon' Puts 700 Million Globally at Risk. Brink News, Marsh & McLennan Companies. Available at: https://www.brinknews.com/de-risking-phenomenon-puts-700-million-globally-at-risk/

Androulaki, E., Barger, A. et al. (2018). Hyperledger fabric: a distributed operating system for permissioned blockchains. In the conference proceedings of the 13th EuroSys Conference (EuroSys '18). Association for Computing Machinery, New York, NY, USA, Article 30, pp:1-15. DOI:https://doi.org/10.1145/3190508.3190538. Available at: https://arxiv.org/pdf/1801.10228.pdf

Antonopoulos, A. M. (2017). Mastering Bitcoin: Programming the Open Blockchain. Amazon Media EU S.à r.l., 410 p.

Babatunde, R. O., & Martinetti, E. C. (2010). Impact of remittances on food security and nutrition in rural Nigeria. Unpublished manuscript, Center for International Cooperation and Development, research paper. Available at: https://www.eadi.org/publications/publication_41344/

Back, A. (2002). Hashcash - Amortizable Publicly Auditable Cost-Functions. 7 p. Available at: http://www.hashcash.org/papers/amortizable.pdf

Baird, L. (2016). The Swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance, 28 p. Available at: https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

Baliga, A. (2017). Understanding Blockchain Consensus Models - Persistent Systems. Available at: https://www.linkedin.com/pulse/understanding-blockchain-consensus-models-arati-baliga

BitcoinWiki. (2018). Available at: https://en.bitcoinwiki.org/wiki/PBFT

Bit-Mari. (2019). BitMari First Remittance Transaction on the Lightning Network, BitHub Africa. Available at: https://bithub.africa/bitmari-first-remittance-transaction-on-the-bitcoin-lightning-network/

Blockchain4innovation. (2017). Blockchain: cos'è, come funziona e gli ambiti applicativi in Italia. Blockchain4innovation. Available at: https://www.blockchain4innovation.it/esperti/blockchain-perche-e-cosi-importante/

BlockChain.com. (2018). Blockchain unconfirmed Transactions. Available at: https://www.blockchain.com/fr/btc/unconfirmed-transactions

Bouvier, P. (2015). Distributed Ledgers Part II: Clearing, Settlements & Legal frameworks. Available at: https://finiculture.com/distributed-ledgers-part-ii-clearing-settlements-and-legal-frameworks/

Buterin, V. (2013). Ethereum white paper. GitHub repository, 22-23. Available at: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper

Buterin, V. (2014). Ethereum: A next-generation smart contract and decentralized application platform. 7. Available at: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

Buterin, V. (2016). Gas cost changes for IO-heavy operations. Ethereum Improvement Proposal. Available at: https://eips.ethereum.org/EIPS/eip-150

Cachin, C., Kursawe, K., and Shoup, V. (2000). Random Oracles in Constantipole: ractical asynchronous Byzantine agreement using cryptography. In conference proceedings of the 19th annual ACM symposium on Principles of distributed computing, Portland, USA, pp: 123-132).

Chain.com. (2014). Chain Protocol Whitepaper. Available at: https://crypto.com/images/chain_technical_whitepaper.pdf

Chan, R. (2016). Consensus Mechanisms used in Blockchain. LinkedIn. Available at: https://www.linkedin.com/pulse/consensus-mechanisms-used-blockchain-ronald-chan

Consensys. (2019). Ethereum Smart Contract Best Practices. Available at: https://consensys.github.io/smart-contract-best-practices/about/reviewers/

Cook, S. (2019). Four Japanese Banks Including the Seven Bank, Joins MoneyTap. CryptoNewsZ. Available at: https://0xfintech.com/2019090466306.html

Cuen, L. (2018). Crypto Startup Wala Is Reaching Africans with Ethereum Micropayments. coindesk. Available at: https://www.coindesk.com/african-startup-ethereums-scaling-issues

Dai, W. (1992). B money. Available at: http://www.weidai.com/bmoney.txt

Demeyer, S. (2017). Research Methods in Computer Science. Proceedings of the IEEE 27th International Conference on Software Maintenance, ICSM 2011, Williamsburg, VA, USA, September 25-30, 2011, doi: 10.1109/ICSM.2011.6080841.

Digiconomist. (2018). Available at: https://digiconomist.net/Bitcoin-energy-consumption

Dingman, W., Cohen, A. et al. (2019). Defects and Vulnera bilities in Smart Contracts, a Classification using the NIST Bugs Framework. International Journal of Networked and Distributed Computing, 7(3):121-132. Available at: https://www.atlantis-press.com/journals/ijndc/125913574

Dwork, C., & Naor, M. (1992). Pricing via processing or combatting junk mail. In 12th Annual International Cryptology Conference, pp: 139-147.

Engelmann, F., Kopp, H. et al. (2017). Towards an economic analysis of routing in payment channel networks. In the conference proceedings of SERIAL'17: ScalablE and Resilient InfrAstructures for distributed Ledgers, December 11–15, 2017, https://doi.org/10.1145/3152824.3152826 .

Etharemit. (2019). Available at: https://etharemit.org/

Ethereum. (2016). Patricia Tree. Available at: https://github.com/ethereum/wiki/wiki/Patricia-Tree

Ethereum. (2018). Ethash Design Rationale revision 18. Available at: https://github.com/ethereum/wiki/wiki/Ethash-Design-Rationale

Everex. (2019). Everex Money Transfer Service. Available at: https://everex.io

F. A. T. F. (2015). FATF takes action to tackle de-risking. Available at: http://www.fatfgafi.org/publications/fatfrecommendations/documents/fatf-action-to-tackle-derisking.html

Freedomnode. (2017). How Bitcoin Works, Freedomnode by Mario Dian. Available at: https://freedomnode.com/guides/17/how-bitcoin-works

Gerard, D. (2017). Attack of the 50 Foot Blockchain. Amazon Media EU S.à r.l. , 182 p. Available at: https://davidgerard.co.uk/blockchain/

Gilbert, S., and Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. Acm Sigact News, 33(2):51-59. Available at: https://users.ece.cmu.edu/~adrian/731-sp04/readings/GL-cap.pdf

Gomedici. (2015). 11 Money Transfert Companies Using Blockchain Technology, MEDICI Global, Inc. Available at: https://gomedici.com/11-money-transfer-companies-using-blockchain-technology-2/

Grandison, T., & Sloman, M. (2003). Specifying and analysing trust for internet applications. Proceedings of the 2nd IFIP Conference on e-Commerce, e-Business, e-Government, Springer I3e2002, Lisbon, pp. 145-157. Available at: http://www.doc.ic.ac.uk/~mss/Papers/I3e2002.pdf

GSMA. (2017). Working Paper Guidelines on International Remittances through Mobile Money, Claire Scharwatt and José Sanin, GSMA. Available at: https://www.gsma.com/mobilefordevelopment/wp-content/uploads/2017/09/GSMA-September-2017-Guidelines-On-International-Remittances-via-Mobile-Money-1.pdf

Gura, N. Arun, P. et al. (2004). Comparing elliptic curve cryptography and RSA on 8-bit. International Workshop on Cryptographic Hardware and Embedded Systems CPUs, Cryptographic Hardware and Embedded Systems - CHES 2004, pp: 119-132. Available at: https://link.springer.com/chapter/10.1007/978-3-540-28632-5_9

Holthaus, E. (2017). Bitcoin could cost us our clean-energy future. Grist Magazine, Inc Available at: https://grist.org/article/bitcoin-could-cost-us-our-clean-energy-future/

IFAD. (2016). Sending Money Home: Contributing to the SDGs, one family at a time, International Fund for Agricultural Development (IFAD), 56 p. isbn: 978-92-9072-751-4. Available at: https://www.ifad.org/documents/38714170/39135645/Sending+Money+Home++Contributing+to+the+SDGs%2C+one+family+at+a+time.pdf/c207b5f1-9fef-4877-9315-75463fccfaa7

Jain, A., Arora, S. et al. (2018). Proof of Stake with Casper the Friendly Finality Gadget Protocol for Fair Validation Consensus in Ethereum. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 3(3): 291-298. Available at: http://ijsrcseit.com/paper/CSEIT1831475.pdf

Johnston, C. (2018). Ethereum Ethstats: Learning the Ethereum Blockchain through its metrics. IMTI blog. Available at: https://imti.co/ethereum-ethstats/

Kaminska, I. (2017). The Currency of the Future Has a Settlement Problem. Financial Times. Available at: https://ftalphaville.ft.com/2017/05/17/2188961/the-currency-of-the-future-has-a-settlement-problem/

Kwon, J., & Buchman, E. (2018). Cosmos A Network of Distributed Ledgers. Available at: https://cosmos.network/docs/resources/whitepaper.html

Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine generals problem. ACMTransactions on Programming Languages and Systems, 4(3):382-401. Available at: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

Li, F., Pieńkowski, D., van Moorsel, A., and Smith, C. (2012). A holistic framework for trust in online transactions. International Journal of Management Reviews, 14(1):85-103. Available at: https://doi.org/10.1111/j.1468-2370.2011.00311.x

LightningNetwork. (2017). The Bitcoin Lightning Network: DRAFT Version 0.5.9.2., Poon, J. and Dryja T. Available at: https://lightning.network/lightning-network-paper.pdf

Magas, J. (2019). Buterin Needs Bitcoin Cash: Scaling Ethereum Before Sharding, Casper. cointelegraph. Available at: https://coinspectator.com/news/1444482/buterin-needs-bitcoin-cash-scaling-ethereum-before-sharding-casper

Malit, F. Jr., Al Awad, M. and Naufal, G. (2017). More than a criminal tool: the Hawala system's role as a critical remittance channel for low-income Pakistani migrants in Dubai, Project: GCC-Asia Migration Corridor, doi: 10.33182/rr.v2i2.429. Available at: https://ideas.repec.org/a/mig/remrev/v2y2017i2p63-88.html

Mazieres, D. (2015). The stellar consensus protocol: A federated model for internet-level consensus. Stellar Development Foundation. Available at: http://www.scs.stanford.edu/17au-cs244b/notes/scp.pdf

McCorry, P., Möser, M. et al. (2016). Towards bitcoin payment networks. In the conference proceedings of Part 1 of the 21st Australasian Conference on Information Security and Privacy, vol. 9722, Springer, pp. 57-76. Available at: https://eprint.iacr.org/2016/408.pdf

Mui, L., Mohtashemi, M., & Halberstadt, A. (20 02). Notions of reputation in multi-agents systems: a review. In the conference proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, July 15-19, 2002, Bologna, Italy, pp. 280-287. Available at: https://dl.acm.org/doi/10.1145/544741.544807

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system, 9 p. Available at: https://bitcoin.org/bitcoin.pdf

Narayanan, A., Bonneau, J. et al. (2016). Bitcoin and Cryptocurrency technologies: Princeton University Press, 308 p. Available at: https://www.lopp.net/pdf/princeton_bitcoin_book.pdf

Nikos, P. (2018). Report on the debate regarding EU cash payment limitations. Jo urnal of Financial Crime, 25(1):5-27, doi:10.1108/JFC-06-2017-0058. Available at: http://www.emeraldinsight.com/doi/abs/10.1108/JFC-06-2017-0058

Olmedilla, D., Rana, O. F. et al. (2006). Security and trust issues in semantic grids. In the conference proceedings of the Dagstuhl Seminar, Semantic Grid: The Convergence of Technologies, Volume 05271. Available at: http://drops.dagstuhl.de/opus/volltexte/2006/408

Patrick M. Jost, H. S. S. (no date). The Hawala Alternative Remittance System and its Role in Money Laundering. Financial Crimes Enforcement Network. Available at: https://www.treasury.gov/resource-center/terrorist-illicit-finance/Documents/FinCEN-Hawala-rpt.pdf

Peck, M. E. (2012). Bitcoin: The Cryptoanarchists' Answer to Cash. IEEE spectrum, May 30th . Available at: https://spectrum.ieee.org/computing/software/bitcoin-thecryptoanarchists-answer-to-cash

Peng, Z. Z. (2015). Prologue to "Demystifying Bitcoin/Blockchain". LinkedIn. Available at: https://www.linkedin.com/pulse/demystifying-bitcoinblockchain-how-actually-works-zong-z-peng-cfa

Ponsot, F., & Obegi, B. (2010). Étude de capitalisation des initiatives et mécanismes en matière de transferts de fonds au Mali. European Commission, Centre d'Information et de Gestion des Migrations (CIGEM), Ministère des Maliens de l'Extérieur et de l'Intégration Africaine Mali. Available at: https://www.findevgateway.org/fr/etude-de-cas/2010/03/etude-de-capitalisation-des-initiatives-et-mecanismes-en-matiere-de-transferts

Poon, J., & Dryja, T. (2015). The bitcoin lightning network. 59p. Available at: https://lightning.network/lightning-network-paper.pdf

Popov, S. (2017). Tangle. 28p. Available at: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf

Pozo, A. (2017). Ethereum: Signing and Validating. medium.com. Available at: https://medium.com/@angellopozo/ethereum-signing-and-validating-13a2d7cb0ee3

Prihodko, P., Zhigulin, et al. (2016). Flare: An approach to routing in lightning network. White Paper. Bitfury Group Limited and Olaoluwa Osuntokun, 40p. Available at: https://bitfury.com/content/downloads/whitepaper_flare_an_approach_to_routing_in_lightning_network_7_7_2016.pdf

Raiden. (2019). Micro raiden, micro paiments for Ethereum. Rayden Pulse #11, News from Macch to April 2019. Available at: https://raiden.network/micro.html

Richards, M. (2015). Software architecture patterns, O'Reilly Media, Inc. ISBN: 9781491924242.

Ripple. (2017). Solution Overview. 26 p. Available at: https://ripple.com/files/ripple_solutions_guide.pdf

Romaldini, M. F. (2018). How Is The International Money Transfer Market Evolving? Toptal Finance. Available at: https://www.toptal.com/finance/market-researchanalysts/international-money-transfer

Roos, S., Moreno-Sanchez, P., Kate, A., and Goldberg, I. (2017). Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions. Cornell University. Available at: arXiv:1709.05748v2

Ryan, D. (2018). Calculating Costs in Ethereum Contracts. Hackernoon. Available at: https://hackernoon.com/ether-purchase-power-df40a38c5a2f

Schiener, D. (2015). Voting on the Ethereum Blockchain: An Analysis. Personal Blog discussing Entrepreneurship, Blockchain and Collective Intelligence. Available at: http://schiener.me/2015/voting-on-ethereum-analysis/

Schmid, S. (2017). Balanced Routing in Micropayment Channel Networks, master thesis, Swiss Federal Institute of Technology, Zurich. Available at: https://pub.tik.ee.ethz.ch/students/2017-HS/MA-2017-11.pdf

Selenium. (2019). Available at: https://selenium.dev/

Siriwardena, P. (2017). The Mystery Behind Block Time. medium.com. Available at: https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a

Stackoverflow. (2016). Exact explanation of statistics displayed on https://ethstats.net. Available at: https://stackoverflow.com/questions/39213693/exact-explanation-ofstatistics-displayed-on-https-ethstats-net

Stellar.org. (2015). On Worldwide Consensus, the future is distributed: a summary of the Stellar Consensus Protocol white paper, Stellar Development Foundation, Available at: https://medium.com/a-stellar-journey/on-worldwide-consensus-359e9eb3e949

Stoykov, L., Zhang, K., & Jacobsen, H.-A. (2017). VIBES: fast blockchain simulations for large-scale peer-to-peer networks. In the conference proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos, December 11-15, Las Vegas, USA, pp: 19-20), doi:10.1145/3155016.3155020.

Szabo, N. (1997). Formalizing and securing relationships on public networks. First Monday 2(9). Available at: https://doi.org/10.5210/fm.v2i9.548

Team, T. D. P. C. (2018). The Problem of Digital Asset Fungibility. The Davis Polk Crypto Team. Available at: https://www.finregreform.com/single-post/2018/05/08/davis-polk-blockchain-bulletin-cryptocurrency-dlt-newsletter-3/

United State Treasury. (2019). Available at: https://www.treasury.gov/resourcecenter/faqs/sanctions/pages/faq_other.aspx

Verification, R. (2019). List of solidity security vulnerabilities. Verified smart contract wiki by Daejun Park. Available at: https://github.com/runtimeverification/verified-smart-contracts/wiki/List-of-SecurityVulnerabilitie

Vishnumurthy, V., Chandrakumar, S., and Sirer, E. G. (2003). Karma: A secure economic framework for peer-to-peer resource sharing. In the proceedings of the 1st workshop on Economics of Peer-to-peer Systems, june 5-6, Berkeley, pp: 1-6. Available at: https://www.semanticscholar.org/paper/KARMA-%3A-A-Secure-Economic-Framework-for-Resource-Vishnumurthy-Chandrakumar/8344aa83a09ebf2b379e31ab84b0e052e59d3c57

Vukolić, M. (2015). The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In the proceedings of the International Workshop on Open Problems in Network Security, Lecture notes in Computer Science, Springer, LNCS, volume 9591, pp: 112-125.

Wander, M. (2013). Datei:Bitcoin Block Data.png. Available at: https://de.wikipedia.org/wiki/Datei:Bitcoin_Block_Data.png#metadata

Western-Union. (2019). FX exchanger. Available at: https://onlinefx.westernunion.com/

Wood, G. (2019). Ethereum: a secure decentralized generalised transaction ledger byzantium version 3e36772 - 2019-05-12. Available at: https://ethereum.github.io/yellowpaper/paper.pdf

WorldBank. (2019). Migration and remittances Recent Developments and Outlook migration and development brief, april 2019. World Bank Group, 28 p. Available at: https://www.knomad.org/sites/default/files/2019-04/Migrationanddevelopmentbrief31.pdf

WorldBank. (2017). Migration and Remittances: Recent Developments and Outlook Special Topic: Return Migration and development brief, World Bank Group, 54 p. Available at: http://documents.worldbank.org/curated/en/719531507124177735/Special-topic-return-migration

Xe. (2019). Currency converter. Available at: https://www.xe.com/currencyconverter/

Xu, X., Weber, I., Staples et al. (2017). A Taxonomy of Blockchain-Based Systems for Architecture Design. In the conference proceedings of the 1st IEEE International Conference on Software Architecture, Gothenburg, Sweden, pp. 243-252, doi: 10.1109/ICSA.2017.33.

Yin, A. S. (2017). A Sneak Peak into the Blockchain Ecosystem After DevCon3. medium.com. Available at: https://medium.com/@awasunyin/a-sneak-peak-into-theblockchain-ecosystem-after-devcon3-4f2934822d3

Zhang, K. (2019). Blockchain presentation at École de Technologie Supérieure.

Zhang, K., & Jacobsen, H.-A. (2018). Towards Dependable, Scalable, and Pervasive Distributed Ledgers. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2-6 July 2018, Vienna, Austria, pp. 1337-1346, doi: 10.1109/ICDCS.2018.00134.

Zohar, A. (2015). Bitcoin: under the hood. Communications of the ACM, 58(9):104-113. Available at: https://doi.org/10.1145/2701411