Technical Report

UCAM-CL-TR-929
ISSN 1476-2986

Number 929

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Resilient payment systems

Khaled Baqer

November 2018

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted October 2018 by the author for the degree of Doctor of Philosophy to the University of Cambridge, St. Edmund's College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Resilient payment systems

Khaled Baqer

Summary

There have been decades of attempts to evolve or revolutionise the traditional financial system, but not all such efforts have been transformative or even successful. From Chaum's proposals in the 1980s for private payment systems to micropayments, previous attempts failed to take off for a variety of reasons, including non-existing markets, or issues pertaining to usability, scalability and performance, resilience against failure, and complexity of protocols.

Towards creating more resilient payment systems, we investigated issues related to security engineering in general, and payment systems in particular. We identified that network coverage, central points of failure, and attacks may cripple system performance. The premise of our research is that offline capabilities are required to produce resilience in critical systems.

We focus on issues related to network problems and attacks, system resilience, and scalability by introducing the ability to process payments offline without relying on the availability of network coverage; a lack of network coverage renders some payment services unusable for their customers. Decentralising payment verification, and outsourcing some operations to users, alleviates the burden of contacting centralised systems to process every transaction. Our secondary goal is to minimise the cost of providing payment systems, so providers can cut transaction fees. Moreover, by decentralising payment verification that can be performed offline, we increase system resilience, and seamlessly maintain offline operations until a system is back online. We also use tamper-resistant hardware to tackle usability issues, by minimising cognitive overhead and helping users to correctly handle critical data, minimising the risks of data theft and tampering.

We apply our research towards extending financial inclusion efforts, since the issues discussed above must be solved to extend mobile payments to the poorest demographics. More research is needed to integrate online payments, offline payments, and delay-tolerant networking. This research extends and enhances not only payment systems, but other electronically-enabled services from pay-as-you-go solar panels to agricultural subsidies and payments from aid donors. We hope that this thesis is helpful for researchers, protocol designers, and policy makers interested in creating resilient payment systems by assisting them in financial inclusion efforts.

Acknowledgements

I am grateful to my family for always being supportive at the most critical moments. I could not have done it without them.

I thank my friends for always encouraging me to go even further and always being supportive.

I thank my funders, the Crown Prince's International Scholarship Program (Bahrain), for supporting my academic studies. I'm grateful for all the assistance and encouragement.

I thank my thesis advisor, Ross Anderson, for motivating me to not only be a better computer scientist, but a better scholar as well. His advice was crucial to guide my research, and to aspire to tackle challenging problems. I've learned a lot from him–not only about computer science–and I'll always be grateful for that.

I thank my undergraduate advisor, Mahendran Velauthapillai at Georgetown University, for always motivating me to give my best and encouraging me to pursue my graduate studies. His advice has been invaluable, and I'm grateful for his counsel.

I thank colleagues who have made my time at the Computer Laboratory better and have always been there when most needed: Christian O'Connell, Jeunese Adrienne Payne, Laurent Simon, and Alexander Vetterl. I'm really grateful for all the great discussions and support; they have enriched my experience at the Lab.

Contents

Chapter 1

Introduction

The modern electronic financial system started with the credit card industry in the 1960s and became profitable in the 1980s. In the 1990s, Europay-Mastercard-Visa (EMV) evolved as the main payment protocol and standard, after prolonged disputes by major stakeholders in the payment industry, with a more recent development of contactless payments. In the same period, a surge of innovations occurred, of which PayPal emerged as the winner [5].

Not all attempts to revolutionise the financial system were successful or transformative. From Chaum's proposals in the 1980s for private payment systems to micropayments and then cryptocurrencies, previous attempts failed to garner traction due to non-existing markets (solutions looking for problems), or issues pertaining to usability, scalability and performance, resilience against failure, and complexity of protocols. These topics are the focus of this thesis: we analyse the aforementioned systems to incorporate their strengths, explore failure issues, and attempt to mitigate them.

The main components of information and computer security are confidentiality, integrity, and availability (CIA), the classic categorisation of security engineering issues that must be tackled to secure and harden systems. Although many other categories are necessary for a more thorough approach, we focus specifically in this thesis on availability; the security community has already dedicated considerable effort to system confidentiality and integrity, to preserve critical data at rest and during transit while supporting authentication and authorisation of system requests. However, as discussed by Anderson in "Security Engineering" [5], most businesses worry about system availability because it is a critical component of their profitability: a system is useless if it is not available, no matter what confidentiality and integrity features it provides; it is unlikely to generate profits, and it may lead customers to seek the services of competitors.

This criticality of the network makes Denial-of-Service (DoS) attacks and connectivity failures among the most detrimental for online services. DoS attacks, for example, render online systems incapable of serving non-malicious requests by overwhelming the system with requests that deplete its resources and can cause system crashes. Such attacks can make systems unavailable for prolonged periods of time. While an online service is under DoS attack, it is not uncommon for attackers to demand a "ransom" payment to cease the attack. Other network failures can have a similar effect on system availability: Domain Name System (DNS) can be poisoned with malicious entries to redirect traffic; and Border Gateway Protocol (BGP) routes can be hijacked to intercept network traffic or create network sinkholes for censorship.

It is not difficult to create a payment system given online connectivity to serve requests through a front end, and a back end to process requests (ideally in atomic operations to

balance debit and credit amounts without loss of data in case of interruptions). However, what is interesting for our research is what happens when a payment system experiences network failures. Such failures are not limited to payment systems; there is a lot of research on fault tolerance and delay-tolerant networks (DTNs) that expect failure and plan for it by, for example, load balancing or delayed delivery of data.

We investigate and prepare for system failure to create resilient payment systems. We define system resilience as the ability to handle and recover from failure, and to temporarily degrade system processes if necessary in order to resume full system operations. We posit that network failures are inevitable, and that for systems to be resilient, offline capabilities must be included in the system design to be utilised when connectivity is unavailable.

We see payment systems as critical national infrastructure that must be protected, possibly from non-local adversaries. A nation can attempt to damage another economically by targeting the victim nation's payment networks to render them incapable of processing transactions quickly or at all. Therefore, it is important to deploy resilient payment systems that can withstand prolonged periods of large-scale attacks – attacks that are not limited to a few select servers, but that target an entire payment infrastructure, whether directly or via its underlying data networks.

Towards creating more resilient payment systems, we investigate issues that impact performance and throughput, network issues, and several other weaknesses that effect scalability and resilience. Since we identified that network coverage issues and network attacks may cripple system performance, the premise of our research is that offline capabilities are required to produce resilience features in critical systems. Offline capabilities used to be standard in banking systems until the 1990s, but were progressively phased out to optimise and simplify system designs. We argue that this has gone too far.

The goal of our work is to tackle network coverage and unreliable service problems by processing payments offline reliably and deterministically, and to simplify the user interface for usability. We focus on financial inclusion efforts, since the problems we investigated must be addressed to extend mobile payments to cater for the poorest demographics.

1.1 Thesis outline

- **Chapter 2 (background)** : This chapter introduces the background to the research conducted, and related relevant work. The focus is on traditional financial systems, which include the payment protocol and standard (EMV) and the payment network that processes EMV transactions – the communication infrastructure and the principals involved. We then discuss cryptocurrencies, focusing primarily on Bitcoin. We also discuss mobile payments and micropayments. We finish by synthesising the challenges and research goals.

- **Chapter 3 (the social externalities of trust)**: We highlight the lack of trust and reputation metrics in open-membership non-centralised systems. The lack of such metrics exposes these systems to various network attacks. We investigate how utilising a payment system, with a pseudo-currency, creates useful metrics for building trust and reputation systems. The system is designed to enable social externalities to determine the level of trust in the system.

- **Chapter 4 (Bitcoin stress testing)**: We analyse a spam campaign that occurred in July 2015, which crippled system performance by overwhelming the network

with spam transactions. This resulted in transactions experiencing more delays in processing time, and incurring higher transaction fees to prioritise transactions.

- **Chapter 5 (offline payment protocols)**: We incorporate lessons learned from reviewing existing systems and previous proposals to create an offline payment system, *DigiTally*, that is focused on financial inclusion efforts in developing countries. The goal of DigiTally is to address network coverage and unreliable service problems by processing payments offline, reliably and deterministically. We aim to create these capabilities while simplifying the user interface and prioritising usability, simplicity, and speed. We include the details of the payment protocol created for DigiTally.

- **Chapter 6 (usability evaluation of offline payments)**: We developed a prototype system for DigiTally, which lets users make offline payments by copying short strings of digits from one mobile handset to another. We discuss the lessons learned from field visits and initial evaluation, and present the findings and observations.

- **Chapter 7 (conclusion)**: The final chapter summarises our work and results, and discusses future work and research towards more resilient payment systems.

1.2 Publications and contributions

I list below my contributions, and those of my colleagues, to the publications that are included in this thesis. I also list acknowledgements of funding and feedback.

1. Khaled Baqer and Ross Anderson. "Do you believe in Tinker Bell? The social externalities of trust", in *International Workshop on Security Protocols*, 2015.

   (a) I worked on the protocol design and collaborated with Ross Anderson to create a usable protocol for Tor bridges.

   (b) I thank colleagues Laurent Simon and Stephan Kollmann for discussions regarding anonymity networks.

2. Khaled Baqer, Danny Yuxing Huang, Damon McCoy, and Nicholas Weaver. "Stressing out: Bitcoin "stress testing"", in *Proceedings of the Third Workshop on Bitcoin and Blockchain Research*, 2016.

   (a) I worked on data analysis, as well as feature extraction and clustering.

   (b) Danny Yuxing Huang helped with the tasks above and was responsible for data gathering, and we both collaborated to produce the results. His PhD thesis [63] includes parts of this publication.

   (c) All listed authors collaborated to set the research goals and the issues to explore.

   (d) This work was supported by US National Science Foundation grant CNS-1619620.

3. Khaled Baqer, Johann Bezuidenhoudt, Ross Anderson, and Markus Kuhn. "SMAPs: Short Message Authentication Protocols", in *International Workshop on Security Protocols*, 2016.

(a) I proposed introducing offline capabilities to produce resilience features in payment systems. I drafted the initial protocol and then collaborated with Ross Anderson to produce a research proposal to apply for a grant. Johann Bezuidenhoudt provided valuable feedback to refine the proposal. This was listed in my first-year report.

(b) After the summer of 2015 (working on the Bitcoin project discussed above), Ross Anderson and I collaborated with Johann Bezuidenhoudt and Markus Kuhn to produce a more robust protocol to account for various attacks and issues. The result of this collaboration is the publication listed above.

4. Khaled Baqer, Ross Anderson, Jeunese Adrienne Payne, Lorna Mutegi, and Joseph Sevilla. "DigiTally: Piloting offline payments for phones", in *Proceedings of the Thirteenth Symposium on Usable Privacy and Security* (SOUPS), 2017.

   (a) Ross Anderson provided valuable feedback to direct this research and handle the connections required for collaboration.

   (b) I designed and implemented DigiTally as a Java Card applet. I provisioned the overlay SIMs and the phones used in the evaluation.

   (c) Jeunese Adrienne Payne and I designed the study, and I was responsible for making sure the applet measured the metrics required for the evaluation.

   (d) I conducted the pilot study and evaluation in Kenya. Tasks included preparing the study material, session demonstrations and tutorials, answering participant questions, and maintaining the data securely upon collection at the end of the study. Staff members from Strathmore University helped with these tasks.

   (e) I prepared the results for analysis and collaborated with Jeunese Adrienne Payne to prepare and write the results of the research.

   (f) Staff members from Strathmore University helped with administrative tasks, selecting and communicating with participants (Strathmore University students), as well as organising and scheduling events and meetings.

   (g) I thank the following colleagues at the Computer Laboratory, University of Cambridge: Laurent Simon for providing feedback throughout the project; Alastair Beresford, Kat Krol, and Alexander Vetterl for helpful feedback on the SOUPS paper. I also thank colleagues at Strathmore University: Edwin Njeru, Nelson Mutua, and David Mutyethau for their assistance with this project.

   (h) This research was approved by ethics committees at the University of Cambridge and at Strathmore University.

   (i) This work was supported by a grant from the Bill & Melinda Gates Foundation.

5. Ross Anderson and Khaled Baqer. "Reconciling multiple objectives–politics or markets?", in *International Workshop on Security Protocols*, 2017.

   (a) Ross Anderson was responsible for most of the ideas in this paper; I contributed sections regarding payment systems and ideas that are related to DigiTally. We also collaborated to list obstacles that suppress innovation and maintain monopolies.

   (b) This paper is not included in this thesis.

Chapter 2

Background

We discuss in this chapter the background to the research conducted and related work. Our focus is on traditional financial systems, specifically, the Europay-Mastercard-Visa (EMV) payment protocol and standard, and the payment network that processes EMV transactions. We then investigate cryptocurrencies, focusing primarily on Bitcoin. We discuss mobile payments as well as micropayments. With regards to mobile payments, we focus primarily on payments using mobile phones as deployed in Less Developed Countries (LDCs), since our project for offline payments was aimed at mitigating some of the problems experienced in these countries.

These types of payment systems are related to the research we have conducted on creating protocols for processing payments securely, and the need for lightweight protocols to minimise bandwidth consumption.

We highlight common problems that we tackled during our research, including performance bottlenecks and network unreliability. The former is a major shortcoming of cryptocurrencies in general, and Bitcoin in particular. Network unreliability also creates resilience issues for the mobile payment systems deployed in LDCs: network disruptions prevent customers from conducting any transactions through the system operator, which is effectively a single point of failure. This is a real problem, especially where network cellular communication is unreliable and susceptible to censorship by the state, which typically either owns the network providers or has regulatory power.

Network issues also arise with Bitcoin: the unavailability of network coverage means that peers cannot finalise payments because a global ledger entry is required to attest to the finality of a transaction. This, in turn, depends on the availability of candidate transactions to the miners.

2.1 EMV

Europay-Mastercard-Visa (EMV) is a card payment protocol and standard that provides interoperability for Integrated Circuit Cards (ICCs) and Point-of-Sale (PoS) terminals worldwide. Another name for EMV is "Chip-and-PIN": EMV smartcards contain a tamper-resistant chip and, in most countries, require a Personal Identification Number (PIN) to authenticate transactions. EMV is maintained by EMVCo., and the latest specification version 4.3 (released in November 2011) consists of four books (or volumes) [52, 53, 54, 55][1].

[1]EMV books include about 750 pages, and start with about 50 pages of acronyms, definitions, terms, etc. that are common between the four books. The effective total number of pages (actual content

Our analysis is largely based on these books (the second book [53] discusses cryptography in EMV). The following overview includes content that we published at SPW 2017 [7].

2.1.1 EMV overview

Card payment networks emerged from the local innovation of Diners' Club in New York in 1950. Both credit and debit cards were first offered by single banks to local customers, then networked together, first nationally and then internationally. The protocol history really starts with the first cash machines fielded by Barclays in 1967. Early ATMs used ad-hoc mechanisms for PIN security but rapidly converged on a methodology pioneered by IBM. A bank could give each customer a PIN generated by encrypting their account number with a secret PIN Derivation Key, available in all bank ATMs and also to its central mainframe.

This was adapted to provide dual control so that no member of bank staff would ever see any PIN other than their own, with a view to both minimising insider fraud and defending the bank against claims from defrauded customers (or those who pretended to be). The mechanism was to embed the cryptographic keys in tamper-resistant hardware, both in the ATM and in a Hardware Security Module (HSM) at the bank. The cards and PINs sent to customers could be sent from different data centres, or from the same centre on different days; and likewise the pairs of keys used to initialise the HSMs in ATMs [5].

The next step was to extend the communications to support interbank networking, a project driven by Visa and other card brands in the early 1980s. Further features were added, such as allowing customers to choose their own PINs. This allowed convenient worldwide ATM networking, but, in due course, problems started to appear: the process had become so complex that attacks started to emerge based on careless design [4] and feature interaction [26]. No sooner had the worst of these attacks been mitigated than the banks rolled out EMV, adding a further layer of complexity and more attacks followed [123].

The card payment system evolved from a single-bank system to an ecosystem, and assurance failed because of a combination of bloat, feature interaction, and institutional economics. As noted in [93], the implementation of EMV in Britain involves thousands of pages of specifications, as the cost of building the new EMV ecosystem was accepting backwards compatibility with numerous legacy systems. The combination of features that led, for example, to the No-PIN attack (discussed in Section 2.1.5) is scattered around this corpus. The bank brands building the ecosystem had insufficient incentive to insist on formal verification, or even adversarial review. The member banks also have conflicts of interest internally [94]: an acquiring bank that uses a less secure PIN entry device increases the risk of fraud across all local card issuers, yet it is unlikely to face more than a few percent of this fraud itself. Ultimately, the limits to security are about governance.

The move from magnetic-strip-based ATMs and PoS transactions to the smartcard-based EMV involved pilots in the early 1990s, standardisation work in the late 1990s, and, finally, a roll-out in 2002, which is still not complete. This roll-out has involved substantial local customisation, and incremental changes are still being made to support new services and patch up vulnerabilities that get exploited. Even there, it can be slow; it took UK banks over five years to block the No-PIN attack [93].

specifying the EMV protocol) is about 550. This does not include specifications for contactless payments.

2.1.2 Transaction processing

The principals in EMV transactions are the *customer* (payer, also called *cardholder*) who has an EMV-compliant card issued by a bank (the *issuer*), a *merchant* (payee) who normally has a PoS terminal that accepts the customer's transaction and processes the payment through the merchant's *acquirer* bank. Cards contain keys that are also stored in the bank's HSMs, which are physical tamper-resistant crypto-processors. The symmetric keys, derived from the bank's master key, are normally changed during the card personalisation process. In the following sections, the notation used is as follows: Card (C), merchant's PoS Terminal (T), and Issuer (I).

2.1.2.1 Card authentication

The EMV protocol specifies three methods for card authentication[2]:

- **Static Data Authentication (SDA):** The card contains a symmetric key used to create a Message Authentication Code (MAC) of the transaction data, called a Transaction Certificate (TC). The terminal must be online to contact the bank, which will verify and authenticate transactions, since the terminal does not contain the symmetric key necessary to authenticate the MAC.
- **Dynamic Data Authentication (DDA):** Asymmetric cryptography is used in DDA to provide more capabilities than SDA. This involves a challenge-response exchange in which the terminal sends a nonce that the card signs and returns, thereby proving that it knows the private key corresponding to the public key in its certificate [116]. However, the actual transaction is not tied to the card authentication process, and the transaction process falls back to using symmetric keys. Thus, the card details are authenticated, but the terminal cannot prove that the transaction was created by the given card (the transaction itself is not authenticated).
- **Combined Data Authentication (CDA):** This method is similar to DDA, with the additional step that the card signs the generated TC, which is the MAC of the transaction, to associate the transaction data with the card. Thus, both the card and transaction details are authenticated using digital signatures.

2.1.2.2 Cardholder verification

This process verifies the cardholder's PIN as entered into the terminal, or a signature might be required instead of a PIN[3]. Note that it is also possible that the terminal does not require verification (no PIN or signature required), which is normally the case for unattended terminals, such as parking meters and vending machines, and for low-value contactless transactions. If a PIN is required, the card sends the PIN counter to the terminal to detect brute force attacks designed to guess the PIN. The PIN is entered by the cardholder and relayed to the card by the terminal. There are encrypted and unencrypted methods to relay the PIN from terminal to card, with encrypted communication requiring DDA or CDA capable cards; it is frequently assumed that the communication between

[2]See [116] for more details.

[3]A signature might be the verification of choice for EMV in the United States with the deployment of EMV-compliant terminals still being finalised.

the terminal and card is not intercepted, which is a critical error that paves the way for various attacks, discussed later. The card replies with a response to the PIN provided by the terminal: either a "PIN OK" message (bytes 0x9000), or a "PIN not OK" message along with the number of trials x remaining (bytes 0x63Cx). This is known as the "offline PIN" verification method; the card itself authenticates the PIN without contacting the Issuer. These messages are central to discussing Man-in-The-Middle (MiTM) attacks on EMV's "offline PIN" verification. The process is outlined below:

$$\begin{array}{lll} C & \to T: & \mathsf{PIN}\ \mathit{counter} \\ T & \to C: & \mathsf{PIN} \\ C & \to T: & \mathsf{PIN\ OK}\ (\mathsf{0x9000})\ \text{or}\ \mathsf{PIN\ Not\ OK}\ (\mathsf{0x63C}x) \end{array}$$

Alternatively, using "online PIN" verification, the PIN is encrypted at the terminal, and the encrypted PIN is sent to the issuer for authentication (the card is not involved in "online PIN" verification). This is the typical transaction flow in ATMs.

2.1.2.3 Transaction authorisation

The terminal sends the transaction details to the card, including the transaction value, currency, timestamp, nonce (which is also called the Unpredictable Number (UN), since it should be random), and other transaction verification data. The card generates a MAC on the data received from the terminal, as well as a counter value stored in the card, called the Application Transaction Counter (ATC), which is incremented for each transaction, and, optionally, the Issuer Application Data (IAD), which contains proprietary data that must be sent from the card to the issuer. The MAC, ATC, and IAD are collectively called the Authorisation ReQuest Cryptogram (ARQC). The terminal then relays to the issuer the transaction data, and the ARQC obtained from the card.

The issuer uses the transaction data received by the terminal to generate a new MAC that must match the MAC contained in the received ARQC. The issuer performs back-end checks to verify that the transaction can be executed, e.g., verifying that the cardholder's account contains enough funds and that the card is not blacklisted. The issuer generates a response based on the checks performed, and sends to the terminal an Authorisation Response Code (ARC) and an Authorisation ResPonse Cryptogram (ARPC) which is a MAC of ARQC $\oplus$ ARC. The terminal uses the ARC value to determine whether to accept or reject the transaction, and passes ARPC and ARC to the card, which generates the same ARPC value using the ARC relayed by the terminal. The card generates a Transaction Certificate (TC), which consists of ATC, IAD, and a MAC of ARC, transaction data, ATC and IAD. The terminal sends the TC value to the issuer, which is used for processing payments.

$$\begin{array}{lll} T & \to C: & \mathit{data} = (\mathit{value}, \mathit{currency}, \mathit{timestamp}, \text{UN}) \\ C & \to T: & \text{ARQC} = \big[\text{ATC, IAD, } \mathsf{MAC}(\mathit{data}, \text{ ATC, IAD})\big] \\ T & \to I: & \mathit{data}, \text{ ARQC} \\ I & \to T: & \text{ARC, ARPC} = \big[\mathsf{MAC}(\text{ARQC} \oplus \text{ARC})\big] \\ T & \to C: & \text{ARC, ARPC} \\ C & \to T: & \text{TC} = \big[\text{ATC, IAD, } \mathsf{MAC}(\text{ARC, } \mathit{data}, \text{ ATC, IAD})\big] \\ T & \to I: & \text{TC} \end{array}$$

2.1.3 Strengths

The main strengths of EMV can be summarised as follows:

1. **Offline capabilities**: EMV transactions can be processed online and offline. In an offline transaction, the card and merchant terminal exchange data and decide whether a transaction should be accepted based on issuer-set risk parameters and configuration. This adds critical resilience features that are necessary when network connectivity is unavailable. Related to this thesis, this is one of the most important strengths of any payment system we reviewed.

2. **Dedicated and tamper-resistant hardware**: EMV implementations provide tamper-resistant hardware for customers (cards), merchants (PoS terminals), and banks (HSMs). Smartcards are offline dedicated hardware devices used to authorise EMV transactions, which are difficult to compromise because of multiple tampering countermeasures, and are only used for payment processing. This trusted hardware guarantees the correctness of the EMV protocol, by protecting the keys used to generate MACs and signatures. Possible issues that undermine the security and trust assumptions are discussed below.

3. **Dedicated network and system resilience**: EMV utilises a dedicated network for clearing transactions. This network is not built on top of the Internet and is less vulnerable to Distributed-Denial-of-Service (DDoS) attacks; this provides resilience for network connections. However, we discuss in the following section issues regarding points of failure in the network.

4. **Verification and auditability**: The cryptography used in EMV provides the ability to create audit logs to verify transactions, with some caveats discussed later.

5. **Usability**: Familiarity and convenience are some of EMV's strong points. Customers are familiar with using bank cards, finding the transaction speed acceptable. The use of trusted hardware to perform critical operations also minimises errors and, therefore, increases usability.

2.1.4 Weaknesses

We discuss below issues that compromise the reliability of EMV or undermine the security guarantees provided by the trusted hardware. We focus primarily on issues around system resilience.

1. **Complexity**: The EMV standard has grown substantially with added features. Optimisations [4] and feature interaction [26] have led to various attacks, discussed later. Moreover, substantial local customisation and new services open up the possibility of lurking vulnerabilities.

2. **Compromised hardware and lack of trusted display**: Although trusted hardware makes it more difficult to compromise the security guarantees and assumptions in the EMV system, there are issues that undermine such guarantees and the assumptions on which they rest. First, the customer does not control the PIN-entry device, which can compromise the transaction through, e.g., skimmers installed on

the terminal. Second, because of the lack of a trusted display, the customer cannot be sure they are approving the correct transaction details, since the terminal may pass any data to the card for signature.

3. **Points of failure**: Although a dedicated network makes the EMV payment network less vulnerable to attacks, an authenticated transaction must still be relayed to an Issuer through the payment processor's network (Visa or MasterCard) in order to determine if the customer has available funds, unless the value of the transaction is small. Yet on 1 June 2018, Visa's network and infrastructure experienced "hardware failure" and started rejecting EMV transactions. Although Visa can normally process thousands of transactions per second (24,000 transactions per second [115], and even more in peak shopping periods such as Christmas), the network failed to process payments by customers in Europe [119].

4. **Maintenance and backwards compatibility**: The Internet carries transaction data when a customer purchases goods or services through an online retailer, and again from online merchants and retailers to payment processors, where they are ultimately processed using the EMV protocol. The network infrastructure has to support both upgrades and backwards compatibility while safeguarding the integrity of the process as well as protecting the customer's information. This creates conflicts. Upgrading systems to use more recent and secure standards must be carefully planned since upgrades may break backwards compatibility. For example, since the end of June 2018, the encryption used to protect transactions must comply with newer versions of TLS, yet older devices (e.g., older smartphone models running unsupported operating systems) may be rendered incapable of transmitting transaction details since they only support older versions of TLS, or what the PCI Security Standards Council calls "early TLS" (TLS 1.0 must not be used after June 2018) [61].

5. **Secure transmission of transaction data**: Related to the point above, online merchants incur non-negligible costs of securely storing, processing, and transmitting payment information to be validated through the payment processor. The protocols used to secure the transmission create various possible failure points, especially if the implementations do not conform to known protocols and standards.

2.1.5 Attacks

EMV weaknesses are largely due to implementation optimisations, rather than a result of the protocol design itself. These optimisations can result in exploits. We provide summaries of attacks that demonstrate how implementation errors lead to broken deployments:

1. **No-PIN attack** [93]: As discussed in Section 2.1.2.2, a terminal sends a message to the card to authenticate the PIN entered by the customer. The attacker executes a Man-in-The-Middle (MiTM) attack to remove the PIN from the message, so that the card thinks it is doing a chip-and-signature transaction. The card then sends a response to the terminal, which is interpreted as indicating the PIN was correct, even if it was not. This MiTM attack is performed in the cardholder verification stage by intercepting the query sent from the card to the terminal and replying with PIN OK (bytes 0x9000). The card, on the other hand, believes that no PIN was

entered, and it simply assumes that an alternative verification method was used (e.g., signature or no verification).

2. **Relay attack** [48]: This attack captures transaction data at terminal A and relays it to a remote terminal, B. The victim authenticates a transaction thinking they are paying for the items they want to purchase. However, the authorisation the victim provides is for a transaction relayed by the merchant's accomplice for a quite different purchase. This exploits EMV transaction authorisation, discussed in Section 2.1.2.3. Relaying the transaction data from terminal B via a card controlled by an accomplice to the malicious terminal A (controlled by the merchant and presented to the victim) means that the victim will authenticate the remote transaction rather than the transaction displayed to them. The lack of a trusted display means that the victims have no means of authenticating transaction details. This attack requires a low latency channel to transmit the data between the two locations. The technical means to overcome the delay in communication between the card and the remote terminal is crucial for this attack to work, since the EMV protocol defines a timeout to receive responses from either the card or the terminal. More time can be requested to delay timeouts by sending bytes 0x0060 to the terminal from the card, or from the MiTM device between the card and the terminal, to indicate a "more time requested" message; the intended non-malicious use of this message is to allow cardholders more time to enter their PIN, used in the cardholder verification stage.

3. **Pre-play attack** [27]: The EMV protocol specification mandates UNs as nonces, but the implementation is often defective, with terminals relying on counters, timestamps, or other non-random numbers. The card counter does not establish the freshness of the transaction; it is the UN value that indicates to the issuer that a terminal issued a new nonce. However, the issuer relies on the terminal for this, and predictable numbers can be generated on a malicious terminal. The effect is that just as the relay attack can shift transactions in space, so the preplay attack can shift them in time: a malicious terminal can get a card to authenticate a number of transactions, which are submitted for payment later. The lesson learned here is that the entity relying on unpredictable values should be in control of issuing them: the bank should control the generation of random numbers, not the terminal [5]. However, if the UNs were always generated by the issuing bank, the bank would have to be online. Designing EMV so that it works offline presumably led to the decision to allow the merchant terminal to stand in for the bank in UN generation – an example of a resilience versus security tradeoff. Given that this tradeoff costs perhaps a few millions in fraud but facilitated many billions in transactions during network outages, it may have been in principle the right decision, although the implementation could have been better. The risk of preplay attacks might have been mitigated by better tamper-resistance of PIN entry devices or by better online fraud detection systems.

4. **Contactless attack** [51]: EMV does not require that a terminal authenticates itself to the card. This vulnerability is exploited to seek contactless EMV cards using, e.g., a smartphone or a card reader, to request transaction authorisation. The card may remain in the cardholder's pocket or wallet and does not require cardholder verification using PIN entry. This can be seen as a digital pickpocketing attack.

2.2 Bitcoin

The cryptocurrency Bitcoin [95] was announced in late 2008 on the crypto mailing list, and the first reference implementation was provided in January 2009. The creator or creators are unknown: the name Satoshi Nakamoto was used by the entity who created and announced Bitcoin, but this name is widely considered to be a pseudonym. The common feature of cryptocurrencies is that they rely on public key cryptography to create, verify, and process payments (hence the name). Cryptocurrencies surfaced after decades of attempts to establish digital currencies; previous attempts to produce a feasible solution struggled to gain adoption for various reasons including performance, fraud, and patents. Earlier attempts at digital money include Chaum's "Blind signatures for untraceable payments" [33], W. Dai's "B-money" [38], and Chaum *et al.*'s "Untraceable electronic cash" [34]. Chaum's proposals were influential in designing non-bank and privacy-preserving systems using blind signatures. Crypto-anarchists and libertarians had sought digital money free from government control since the 1990s [82, 83, 84].

There are many aspects of Bitcoin that are not discussed in detail here as they are beyond the scope of this research. The following sections focus on the resilience of the system with regards to performance and throughput, network issues, and other issues that effect its scalability. We discuss the system and protocol as designed, and highlight the difference between theory and practice where relevant.

2.2.1 Bitcoin overview

Bitcoin is a decentralised Peer-to-Peer (P2P) payment system. The Bitcoin payment system does not have a bank server to act as a central root of trust; this is one of the main differences between cryptocurrencies and other payment systems. One of the main problems with maintaining central payment servers, and a motivation to create decentralised payment systems, is system resilience: those servers can be vulnerable to system disruption via, e.g., DoS attacks. The promise of a decentralised system is that in order to cripple the entire system many nodes must be neutralised; at least in theory this is what Bitcoin tries to maintain (we discuss this further in Section 2.2.3).

A *bitcoin*[4] is not a physical token or currency. Instead, owning a bitcoin in effect means owning the private key[5] required to spend that bitcoin. Spending a bitcoin involves changing the record of ownership, effectively changing control of a bitcoin from one key to another. To change ownership, a receipt of the modification, created by the last owner, must now record who the new owner is. This receipt is a bitcoin *transaction*: a signed receipt determining the next owner of a certain quantity of bitcoins, signed by the previous owner of those bitcoins. Transactions include *inputs*, which refer to previous unspent transaction outputs (UTXOs) to be used in the current transaction, and *outputs*, which are Bitcoin *addresses*[6].

If A wants to pay B, then A must sign a transaction that 1) proves that A owns all the funds input to that transaction, and 2) declares that B is now the new owner of

[4]We follow the convention of referring to the protocol as *Bitcoin*, the currency and its units as *bitcoin* or BTC.

[5]Bitcoin uses Elliptic Curve Cryptography (ECC).

[6]A Bitcoin address is a 160-bit hash of a public key as well as a checksum, represented in Base58. Addresses can be exchanged out-of-band.

the funds included in that transaction[7]. How can B be sure that A did not spend the same bitcoins to, say, C in a *double-spending attack*? If B accepts a signed transaction from A directly, B could not be sure that A did not cheat by sending another signed transaction to C as well. Previous payment systems used either a central bank server, or hardware tamper-resistance, or both, to make double spending hard. Bitcoin's novelty is that there is neither. Instead, for the principals involved in a transaction (A and B in the example above) to be sure that a transaction is final, the *whole network* must agree on the finality of their transaction. The global ledger – the *blockchain* – which creates the global consensus, records all transactions considered final by the Bitcoin network. This ledger will, by definition, include all the ownership changes and the ledger itself performs the role of a bank server.

Transactions can include *transaction fees*, which are 1) a monetary incentive to encourage transaction processors[8] and nodes in the network to validate and propagate transactions[9], and 2) a tax to demotivate spam (e.g., a high number of low-value transactions). Fees are the unclaimed outputs of a transaction, and the processor of the transaction can claim those outputs by assigning themselves as the new owners. Spam transactions and transaction fees are a focus of Chapter 4.

Losing a record of a Bitcoin transaction is inconsequential; it is ownership of the controlling private key that matters. Therefore, a Bitcoin wallet is no more than a collection of private keys that spend bitcoins. Losing those private keys renders the bitcoins owned by those lost keys non-spendable. If the keys are stolen, then the thief can now spend the bitcoins.

We now explain how global consensus is maintained, as well as the mining process and the performance issues in the Bitcoin protocol. We focus primarily on aspects of Bitcoin that are required to understand the problematic issues we have investigated in our research, which are discussed in detail in Chapter 4.

2.2.1.1 Global ledger and money supply

In a decentralised system, without a known set of bank servers, who creates the global ledger and how is global consensus reached? In Bitcoin, participating nodes create a *block* which is a data structure containing verified transactions. Verification means checking that the signatures are valid using the public keys provided along with the transaction: the public keys must hash to the addresses of the unspent transaction outputs (UTXOs) that are inputs to that transaction. A block also includes a Merkle Tree of the verified transactions. Blocks, which are created roughly every 10 minutes, also include a reward, called the *coinbase transaction*, for creating the block. This special coinbase transaction has no prior history of ownership and is created by the node that produced that block. In effect the node that created the block is paying itself in the block it created, and announcing this reward to the entire network. This is how bitcoins are created and injected into the system[10].

[7] We provide a simplified example here; a transaction may contain multiple inputs that may be controlled by multiple principals so long as they all sign it, and multiple outputs too.

[8] We discuss later the role of transaction processors, called Miners.

[9] Transaction fees prove that the sender is incurring a cost, the assumption being that an economically rational entity would avoid incurring unnecessary fees.

[10] The block reward is halved every four years (approximately), and is now 12.5 BTC (about $87,500 at the current exchange rate).

A new block (b_i) must include a hash of the previous block (b_{i-1}). This chain of block hashes is known as the *blockchain*, and represents the global ledger, where a transaction must be included to be considered final. This blockchain includes all the historical changes of ownership of every bitcoin. The participating nodes not only verify transaction signatures, but also check that each input has not previously been spent. Thus, if a transaction authorising the transfer of funds from one owner to the next is already included in the blockchain, the previous owner cannot reuse the same private key to transfer the funds elsewhere, and any double-spending attempt should be rejected as invalid.

2.2.1.2 Mining and transaction processing

As discussed earlier, a block contains a reward to incentivise the creation of blocks. This reward and process must be controlled, otherwise any node in the network can create blocks at will to claim rewards. The rate of creating blocks in the network must be throttled to avoid overwhelming the network with newly created blocks. Rapid block creation would render global consensus impossible to reach, since there would be no way for a node to determine which new block is the correct one to add to the blockchain.

The ability to create blocks is determined and controlled using a Proof-of-Work (PoW) system that involves solving computationally difficult puzzles. PoW was proposed to fight spam [50] and was used in HashCash [14] as a DoS countermeasure. It is used in Bitcoin to ensure that a node needs to do substantial work to find a hash collision. This entails hashing the current block, which may include verified transactions, and must include the hash digest of the previous block, in order to find a hash that includes d zeros in the prefix, where d is referred to as the *difficulty*. The cryptographic strength of the hash function means that there is no faster way known of doing this than iterating the process with different random seed values (nonces) in the block until a correct collision is produced. This involves on average 2^{d-1} hash computations for a difficulty level of d.

Finding such a solution entitles the successful principal to claim a block reward. This computationally intensive process of verifying transactions and creating blocks is called *mining*; *miners* create blocks and claim rewards. Once a miner finds a new block, it is broadcast to the network, with an average propagation time of about 13 seconds [39]; thereupon the other miners start using it as the basis for finding the next block. The Bitcoin protocol recognises the longest branch of the blockchain as the correct one, creating an incentive for consensus.

Bitcoin includes more monetary incentives: miners can claim transaction fees, explained in Section 2.2.1. Since a transaction remains non-final until it is included in a block, as discussed in Section 2.2.1.1, and miners can claim any unclaimed transaction outputs as transaction fees, they can select the transactions offering the highest fees to embed in a new block. They then work on the computational puzzle to create a valid block: once one miner wins this race, and their solution is accepted on the blockchain by a majority of other miners, that block will be built on and the claim to the block reward and transaction fees will become bankable.

The intention behind mining, as discussed in the original Bitcoin paper [95], may have been naive. That paper includes the term "one-CPU-one-vote", which primes the reader to appreciate the democratic values of the proposed system: the idea is that any one entity would control a CPU and would, therefore, have a chance to contribute to the network and potentially earn block rewards. However, the mining ecosystem has long progressed from using CPUs for mining: the ability to parallelise hash computations led first to a

move to utilise the superior capabilities of GPUs, then to field-programmable gate arrays (FPGAs), and finally Application-Specific Integrated Circuits (ASICs). ASICs produce better performance and now perform all the hashing of consequence.

Another interesting development in the mining ecosystem is the cooperation between miners to create mining pools. Miners pool their resources and a pool operator sends all miners a block to hash repeatedly until a valid solution is produced, paying miners based on their contribution[11]. This has resulted in the centralisation of mining effort, and fierce competition between mining pools to claim block rewards by adding more hardware: the current *network hash-rate* (the rate at which the network can generate hashes collectively), is 35-40 tera hashes per second[12].

It is possible that multiple miners produce multiple valid blocks b_i at epoch i. This would prevent global consensus since listening nodes will not be able to determine which block to add to the blockchain: a temporary fork can occur where the network splits by adopting multiple versions of the ledger, and this violates global consensus. Bitcoin resolves this issue by dictating that the fork that invested the most work, as determined by inspecting the blocks and their PoW, should be chosen as the correct blockchain. In practice, this scenario may not occur often. If it does occur, global consensus would be delayed until the fork is resolved. This opens up the system to disruption attacks.

2.2.1.3 Disruption attacks

1. **51% attack**: If a miner creates a longer blockchain than the one in current use, then that miner can disrupt consensus as the longest blockchain is assumed to be the authoritative global ledger. This would cause previous transactions to be invalidated and considered non-final in the new alternative blockchain, and the attacking miner would claim rewards for all blocks included in the new blockchain, invalidating previous block rewards. Thus, the more difficult it is to create a block, the more difficult it becomes for an attacker to disrupt consensus and create a longer blockchain. Forks are problematic not least because they undermine Bitcoin's finality and irreversibility claims. This is exacerbated by the fact that Bitcoin lacks any consumer protection or dispute resolution mechanisms. Nakamoto's idea was that any coalition of miners controlling a minority of computational power (hash-rate) should not be able to out-mine a group of correctly-operating majority miners to create an alternative blockchain [95, 21]. Three mining pools, at the time of writing, could collude to control more than 51% of the network hash-rate: BTC.com controls 27.9%, AntPool 12.4%, and SlushPool 11.1%. This undermines the claim that Bitcoin is decentralised: miners controlling the majority of the hash-rate have a higher chance of creating blocks, and therefore have a good chance to out-mine everybody else and dictate what transactions are included in the blockchain.

2. **Selfish mining**: There are circumstances in which miners might conduct an attack and disrupt or take over a blockchain with less than 51% of the hash rate. Active adversaries can monitor the network and strategically broadcast their blocks to claim rewards and disrupt competitors' capabilities in what is called a selfish-mining attack [56]. An adversary broadcasts blocks when other miners are likely to be

[11] There are a few variations of how mining pools work; we included a simplification of the most basic and earliest mining pool schemes.

[12] See `https://www.blockchain.info/charts/hash-rate`.

close to catching up with the adversary, and does not broadcast anything otherwise. Broadcasting a longer blockchain would invalidate non-malicious miners' blocks and rewards, making their mining operations unprofitable. Given that miners invest their hash rate wherever the return is highest, this incentivises economically-rational miners to join an adversary who mounts a successful selfish-mining attack. Moreover, it is possible for an adversary who controls many IP addresses to connect to nodes in the Bitcoin network to control what data is relayed to those victim nodes, in what is called an "eclipse attack" [62]. Using an eclipse attack, the selfish-mining strategy can be more effective: an adversary who is well connected in the network can control which blocks are propagated in the network, censoring blocks created by other miners, and increase the chances of a successful selfish-mining attack[13].

3. **Double-spending attack**: Using the ability to disrupt global consensus, as explained above, a computationally powerful adversary can choose to double-spend their funds. This can happen when A pays B in transaction tx_B included in block b_i and B waits for and accepts b_i as final (at this point, B and A exchange goods or services based on the assumption that tx_B is final and irreversible). However, after the exchange, A creates two chained blocks b_i' and b_{i+1} where either of those blocks includes a transaction that conflicts with tx_B, that pays someone else in tx_C using the same funds (or indeed A can reclaim the funds in tx_A). By Bitcoin's rules, the longer blockchain (b_i' and b_{i+1}) will be chosen as the correct ledger, while b_i will be discarded, rendering tx_B invalidated and non-final. Moreover, tx_B would be considered a double-spend attempt, since it conflicts with transaction tx_A or tx_C that is included in either b_i' or b_{i+1}. Therefore, B would provide their goods or services without being able to claim funds in tx_B [13, 72, 56].

The cost of disruption attacks is the computational cost of creating an alternative chain.

2.2.1.4 Security economics and incentives

The Bitcoin community decided, arbitrarily, to wait for six blocks in order to mitigate the risk of a fork happening in the blockchain that could compromise the global consensus (six blocks are created in approximately an hour). Regarding the double-spending attack explained above, this means that B would wait for five more blocks created after b_i before considering a transaction as final. This increases the difficulty of A disrupting the consensus by creating an alternative blockchain that is longer than six blocks.

In practice, users may not wait for six blocks to be created, and instead accept a transaction as valid and final once it is included in a new block. These users are then susceptible to double-spending attempts that can mount disruption attacks, discussed earlier. Even more problematic, if users accept transactions directly without waiting for a transaction to be included in a block, a double-spend attack is easier to mount: a malicious user can broadcast an alternative transaction, and if it is included in a block, it would render the original transaction invalid. It is possible to offer higher fees in the alternative transaction to incentivise miners to prioritise it for inclusion. Furthermore, it is possible that users perform off-chain transactions: creating and accepting transactions using protocols that bootstrap payment channels on the blockchain that create direct

[13] See [30] for a discussion about combining eclipse and selfish-mining attacks.

relationships to spend bitcoins, with the ability to close the payment channel and settle the balances by including the latest valid transaction on the blockchain [37].

The security-economics argument against malicious disruptions is that an economically-rational miner would use their computational capabilities to operate correctly and mine blocks to claim rewards. A counter argument is that this would not prevent an adversary whose incentives are not purely economic, or if the double-spend attack involves sums that are much larger than what a miner would be able to claim in rewards and fees. Another argument against malicious behaviour is that an adversary who operates this way would undermine the trust in the network, leading to a collapse of the value of bitcoin, which cuts the rewards a malicious miner could obtain in the future. The same counter argument of an adversary who is not economically rational applies.

To mitigate disruptions in global consensus, given that the longest blockchain is considered as the global ledger and all alternatives (forks) are discarded, a miner who is economically rational will aim to maximise their profits as follows:

1. **Broadcast blocks widely**: this maximises the miner's chances that their block will be added to the blockchain. All miners then add that block to the blockchain and start working on the next block. Block propagation takes 13 seconds on average [39], so a miner (or mining pool) will want to have their block accepted as correct before another valid block is propagated in the network.

2. **Listen for block broadcasts and react immediately**: a miner will stop working on their own version of the block if a valid block for that epoch is received. This demotivates working on alternative versions of the blockchain (forks).

If a miner does not follow these two rules, they risk wasting computational effort producing a block that will not be adopted by the majority of the network, if the network has already progressed towards creating the next block. These incentives create the required stabilising factors to produce a consistent global ledger and achieve consensus in the network by demotivating forks and selfish behaviour [74, 106]. The monetary incentives are crucial to system stability when a competitive protocol is used to create consensus.

2.2.2 Bitcoin's backlog: Mempool and UTXO

Transactions are broadcast in the network to other peers for inclusion in a new block. Participating nodes listen on the Bitcoin P2P network[14] for broadcasts of new hashes of transactions and blocks. The nodes then issue requests to their peers to retrieve previously unknown data: nodes send `getblock` and `getrawtransaction` to retrieve new blocks and transactions, respectively. The newly received data are validated, and the transactions are added to the node's local memory in what is called the *Mempool*. A transaction stays in the Mempool until it is finally included in a block. However, memory pressure (or other algorithms used independently by nodes), may evict a transaction from an individual node's Mempool before that transaction is included in the blockchain.

Bitcoin transactions, as discussed earlier, consist of inputs and outputs: the inputs reference Unspent Transaction Outputs (UTXO): nodes who validate transactions sent in the network, especially miners who validate transactions to be included in a newly created block, keep track of UTXOs to quickly verify that a transaction includes unspent funds,

[14]By default, Bitcoin uses TCP/8333 for communication.

as well as checking the signature. Due to the P2P broadcast of transactions, the Mempool and UTXO sets may not be the same for all nodes in the network. There is no global consensus for the Mempool and UTXO; it depends on a given node's vantage point and what it receives in the P2P network.

Since transactions remain in a participating node's memory until they are finalised, this places resource pressure on the node to maintain correct operations. The pressure decreases each time miners include transactions in a new block: only then will all nodes evict transactions from their Mempool. However, a backlog of transactions remains in memory as the network works to include transactions: at the time of writing, the Mempool alone is 3 GB in size, composed of distinct and previously unseen transactions[15].

Requiring more resources as the Mempool and backlog grow could open up a node to a targeted attack to deplete its available resources, possibly leading to the node crashing if it cannot keep up with resource pressure.

More problematic for system performance is that miners can create *empty blocks*, to claim block rewards without including any transactions (besides the coinbase transaction needed to claim the reward). This causes the blockchain to grow with empty blocks, as miners receive rewards without alleviating the backlog in the Mempool. Miners have no incentive to process and validate transactions other than transaction fees, which tend to be very small except at times of congestion. In fact, a rational miner could have a negative incentive to include transactions in blocks, since including any invalid transaction would result in their block being rejected.

The Bitcoin backlog is a central focus of the spam campaign we examined in 2015. We provide the details of our analysis in Chapter 4.

2.2.3 Novel features

One of the main novelties in Bitcoin is its attempt to produce robust global consensus of a variable number of anonymous nodes without a central server. Previous protocols for achieving distributed consensus deploy voting procedures, e.g., Paxos [75] and PBFT [32], in order to replicate the ledger among a known set of servers (closed membership). The main problem with such systems is that the known set of servers are vulnerable to disruption, such as a DoS attack. Another problem is that voting protocols do not scale well with an increase in the number of servers.

Furthermore, standard voting protocols for distributed consensus assume a known set of validators (n), of which a threshold (k) must be available and operating correctly to make progress and create consensus, despite a number (f) of faulty or compromised validators. To make things more efficient, a *leader* is chosen in a given epoch (or remains as a leader for a few epochs) to manage tasks and messages sent by validators. Compromising the current leader cripples progress and severely degrades system efficacy.

In contrast, Bitcoin promised, for the first time, a *leaderless* consensus protocol, without the k-of-n threshold requirement. This means that the entity (leader) announcing the progress in the network (the next block) is hard to predict among the set of all participating nodes (miners), meaning there is no single point of failure to compromise or DoS. Therefore,

[15]During increased non-malicious activity, or possible DoS attacks, the Mempool can increase substantially. In May 2017, and between November 2017 and January 2018, the Mempool increased to almost 100 GB in size. The latter period witnessed a substantial increase in the price of bitcoins, and may have been due to an increase in speculative trading transactions.

to stop the system from operating correctly, many participating nodes must be neutralised; this increases the work an adversary needs to perform.

In practice, we discussed the aggregation of mining power in mining pools in Section 2.2.1.2; prominent miners and mining pools are well-known in the Bitcoin network, making them vulnerable targets to their competitors (other mining pools) or disruptive adversaries. Despite the possibility of compromise or disruption, the protocol is open to all (open membership): anyone with sufficient computational power can start mining to produce blocks, and maintain progress and consensus in the network. However, the barrier to entry is high. If major mining pools become unavailable (due to a DoS attack, nation-state censorship, or simply working on cryptocurrencies other than Bitcoin), substantial computational power will still be needed to start mining to replace the network hash-rate. Alternatively, the entire network must wait until the difficulty d and hash-rate, discussed in Section 2.2.1.2, is globally recalibrated to reflect the decrease in mining power[16]. This recalibration happens every two weeks, which is a theoretical downtime for Bitcoin if the current computational power suddenly vanished.

The finality of Bitcoin's decentralised consensus comes from the substantial computational power required to mine blocks by solving PoW puzzles. The assumption is that after several blocks are mined, it is substantially more difficult to create an alternative blockchain that would be longer than the currently known one. Thus, it is assumed that miners are incentivised to operate correctly and gain rewards rather than attack the network. Therefore, Bitcoin depends on whether its incentives for non-malicious behaviour produce the necessary stabilising factors to maintain global consensus.

2.2.4 Strengths

The following discussion is not an exhaustive list of Bitcoin's strengths and weaknesses. Rather, we discuss issues that are related to this thesis. Bitcoin strengths can be summarised as follows[17]:

1. **Trusted hardware is not required**: Bitcoin does not involve or require dedicated tamper-resistant hardware. On one hand, this cuts the barrier to entry: users need only download the necessary software (a Bitcoin client implementation) to participate. It also cuts the cost of participation (sending and receiving payments), and increases mobility in the sense of portability of data to other machines. Bitcoin relies on the strength of its protocol, and the mathematics of the cryptography, rather than the protections provided by hardware, such as smart cards or HSMs on which new attacks might be found. On the other hand, Bitcoin implementations can use hardware to resolve usability, complexity, and security issues. For example, offline hardware wallets protect private keys and create transaction signatures within the tamper-resistant device. This prevents the keys from being exposed unprotected on user devices (e.g., smartphones or laptops). However, this comes at a high cost

[16]Recalibration is done through software clients, measuring the rate of block creation in the network and adjusting the acceptable value for d to enforce an average block-creation time of 10 minutes.

[17]Note that we classify the strengths and weaknesses from a perspective of consumer protection and dispute resolution. For example, the finality and irreversibility of a transaction can be viewed as a strength by someone who requires transactions to be finalised immediately, yet from a consumer protection perspective, the ability to reverse transactions to recover from fraud and error is crucial. In Bitcoin, transaction irreversibility is viewed as a feature, but from our more traditional point of view of prioritising consumer protection, it is a bug.

if the hardware wallet is lost or stolen: in most cases, backups and export of the hardware-generated keys are not possible or non-trivial, and losing access to the keys means that the bitcoins controlled by those keys are forever irrecoverable.

2. **Censorship-resistance**: The decentralised consensus protocol means that there is no central entity, or a group of entities, that can be seized, ordered to disclose user information, or freeze user accounts. Even if major miners are unreachable, other miners can take their place and maintain the blockchain. The main assumption here is that there exists no powerful adversary controlling the entire Internet with means to disrupt all communication links upon the detection of Bitcoin traffic.

3. **No single point of failure; consensus without a leader**: Bitcoin's consensus protocol provides a means of reaching consensus without knowing the next epoch's leader, which provides enhanced resilience against network and system disruption. However, in our own research, we show that the network itself is not Bitcoin's vulnerability, but rather it is the limited throughput that can be overwhelmed. We discuss this further in the next section, and provide the details in Chapter 4.

4. **Low transaction fees (for now)**: Transaction fees are relatively low compared with traditional financial systems, we discuss this further in the following section.

5. **Pseudonymity**: Bitcoin uses pseudonymous identities (the hash of the public keys), and no personal information is required to participate in the network. This feature appealed to privacy-conscious users, at least initially. However, this point is also mentioned in Bitcoin's weaknesses, since the level of privacy is not optimal.

2.2.5 Weaknesses

We discuss several issues that hinder scaling the network as more users adopt Bitcoin:

1. **Lack of offline capabilities**: Payments can only be accepted online, because a transaction must be sent to miners who include that transaction in a block in a globally-verifiable public blockchain ledger. Accepting the payment also requires online connectivity; the blockchain declares that a bitcoin's ownership has changed. This enables the recipient to verify, through the network, that they received new bitcoins. There is no means for processing a payment directly between peers in an offline transaction, including peers communicating directly through ad-hoc local channels such as Wi-Fi peering, Bluetooth, NFC, etc., to finalise a payment; a global online consensus must be established. In contrast, EMV can fall back to offline processing, where terminals can accept signed transactions issued by the trusted hardware, to be processed later when system connectivity is restored.

2. **Inefficiency**: The process of verifying transactions and including them in the blockchain needs the equivalent of a supercomputer, due to the current difficulty of solving the computational puzzle. It requires significant power consumption, and someone to pay the bill; and this requirement is only increasing. This inefficient use of power must continue in perpetuity to maintain the blockchain in a competitive environment [76, 77].

3. **Irreversibility and finality**: Once a signed transaction is included in the blockchain, and assuming the current block remains valid, that transaction is irreversible and the funds cannot be reclaimed. Bitcoin proponents think this is a feature rather than a bug: irreversibility prevents fraudulent users from reversing transactions after receiving goods or services, which is a major venue for fraud in EMV-based systems through charge-backs [6, 5]. However, since there is no customer service number to call to try to reverse mistakes, rectifying mistakes is impossible. Furthermore, Bitcoin (and similar cryptocurrencies that require a transaction to be included in a global ledger) experience delayed processing of transactions, compared with EMV transactions, which take seconds to authorise. Although a Bitcoin transaction may also take only seconds to propagate in the network once a payer signs and broadcasts it, it must be included in a block to have effect, and a block is created, on average, every ten minutes. Moreover, for reasons outlined earlier, users are advised to wait until five more blocks are created to prevent forks leading to non-finality of transactions. However, a transaction may not be included in the next block, and remains in limbo until a miner selects it from the Mempool to include it in a new block. This means that the 60-minute processing time, to ensure finality, can be seen as a lower bound, particularly at times of congestion.

4. **Lack of effective privacy**: The blockchain is a public ledger accessible by anyone who cares to download it from the Bitcoin network[18]. Bitcoin protects users' privacy through the use of pseudonyms. This is also considered a feature, yet correlating metadata of a few transactions (timing, value, frequency, etc.) is often an easy way to link a user to a pseudonym. Researchers have already pointed out that Bitcoin has significant problems with regards to privacy [11, 101, 105]. Knowing a user's address is sufficient to investigate the entire blockchain for all purchases made, and funds received, using that address. This can be considered a privacy downgrade when compared to the level of privacy a user of traditional financial services expects: information about bank transactions is only available to the payer, the banks involved, the payee, and authorities such as the police and financial regulators who get access using warrants. Other researchers investigated correlating users' online activity and purchases, revealed through web cookies, with their Bitcoin transactions [60]. With the EU's General Data Protection Regulation (GDPR) which went into effect in May 2018, it will be interesting to follow how that effects issues regarding data control and deletion: data on the blockchain is supposed to be immutable and cannot be deleted upon request.

5. **Scalability and performance issues**: Bitcoin faces scalability issues with regards to increasing the number of transactions included in each block. Each block would significantly increase in size leading to propagation and consensus issues (the blocks will be slower to propagate). There would be an increase in the rate of growth of the blockchain, which must be maintained by miners to verify transactions. It is possible for users who are not miners to use Simplified Payment Verification (SPV) and this was discussed in the original Bitcoin proposal [95], but this comes at a cost of having to trust other nodes to do the mining and thus to verify transactions: Bitcoin promised a decentralised trustless payment system without having to trust super nodes. We discuss in Chapter 4 how Bitcoin's performance can be crippled

[18]The blockchain is currently about 170 GB in size.

through spam campaigns that substantially increase system resource pressure and delay transaction processing.

6. **Transaction fees and a future without block rewards**: Fees are only low for now; when the block rewards are substantially lower than what they are now, mining will not be profitable (unless the value of bitcoin increases substantially), and transaction fees must be increased to cover the amount of computation performed by miners. Bitcoin is bound to lose its major advantage over more expensive money transfer systems. In fact, this is already happening when the value of bitcoin increases (the value and exchange rates are volatile). Bitcoin transaction fees are equal to, if not higher than, traditional payment systems, especially when taking into account the backlog to clear payments, and the higher transaction fees a user must pay in order to prioritise their transactions. The security economics of Bitcoin incentivises correct behaviour through block rewards. However, the Bitcoin protocol dictates that the block reward is halved approximately every four years until the block reward is eliminated. As discussed in [31], a lower block reward may result in an unstable network if miners rely on transaction fees rather than block rewards. Miners would have an incentive to mine blocks in order to maximise transaction fees, rather than to work on the longest blockchain. For reasons discussed in Sections 2.2.1.2 and 2.2.1.3, this can undermine the stabilising factors necessary for Bitcoin's global consensus.

7. **Non-fungibility**: Not all bitcoins are equal; each bitcoin differs in its ownership history. This can be a problem if some bitcoins are blacklisted due to illegal activity. This is not a problem for cash since every coin or note is fungible (serial numbers are usually ignored) [92]. There are proposals to address this non-fungibility issue for cryptocurrencies, and to provide more privacy features [87, 107].

8. **Usability issues and increased risk**: A user must maintain the private keys required to spend her bitcoins. Can we expect users not to lose private keys? Compare this with requiring users to remember their 4-digit PINs for their EMV cards or their online banking passwords; both can be reset if required, whereas losing a private key destroys the value of those bitcoins that are spendable using that key. In practice, however, there are a number of exchanges and so-called *online wallets* that manage bitcoins on behalf of users, just like banks do (except these online wallets operate without any regulatory oversight). Users open an online wallet through a provider by creating an online account protected with only a username and password. This effectively removes the "crypto" in *crypto*currency: the accounts are no longer managed directly by the user, and the online wallet provider must be trusted to manage keys on behalf of users. This leaves the customers with no recourse as there are no consumer protection mechanisms to retrieve the funds if the provider is compromised by hackers or insiders who steal private keys, or the provider absconds with the bitcoins. Moreover, due to unfamiliarity with Bitcoin, users are apprehensive about joining a financial system they do not understand and perceive as risky if not assured by local governments that their funds are secure. This is exacerbated if the average user is only exposed to Bitcoin's misuse for ransomware, illegal online drug sales, and other so-called "dark web" activity.

9. **Barriers to entry and mobility**: Although the barrier to entry to participate in the system (to send and receive payments) is low, converting fiat currency (cash

or transfer from bank accounts) to bitcoins faces many hurdles and the friction is increasing: in June 2018, Wells Fargo was the latest bank to ban purchases of cryptocurrencies using its cards[19]. Users face obstacles to *cash-in* and *cash-out* from the system, i.e., to convert fiat into bitcoins, or bitcoins into fiat, respectively.

10. **Non-compliance with KYC and AML regulations**: Bitcoin's pseudonymity is incompatible with Know-Your-Customer (KYC) and Anti-Money-Laundering (AML) regulation, which requires maintaining accurate and verifiable information regarding identities and ownership of financial assets.

11. **Problems with Bitcoin's consensus**: The ability to mount disruption attacks (e.g., 51% attacks and selfish mining, as discussed in Section 2.2.1.3), can undermine trust in the system's capability to reach consensus on the global ledger. A persistent attack might confuse the network, can result in double-spends, demotivate non-malicious miners since they can lose their block rewards, lead to an exodus from the system resulting from a lack of trust and, ultimately, to the collapse of the value of the currency. This is more problematic if a motivated adversary is not economically rational; the security economics embedded in Bitcoin would then not produce the necessary stabilising factors for a global consensus. Hence, Bitcoin's protocol can be called *doubtful consensus* as pointed out by Laurie in [76, 77]: one can never be sure that someone else is not secretly working on a longer alternative blockchain that can nullify the current blockchain[20].

12. **Network attacks**: Eclipse attacks [62], as discussed in 2.2.1.3, can result in censoring victim nodes' view of the network, and disrupting consensus in the network by preventing block propagation. Moreover, an adversary can hijack network traffic, by advertising new BGP prefixes, to segment the network into non-communicating quorums [12], to achieve similar results to an eclipse attack, or to hijack mining pool traffic (to earn mining pool rewards or to disrupt competitors). In P2P systems, network attacks similar to the ones highlighted here, and with the absence of a global view and coordination to establish trust and reputation metrics, demonstrate that such systems are vulnerable to attack without an authority being able to mitigate the presence of malicious nodes who can disrupt various critical components in the network and remain undetected. We discuss in Chapter 3 a scheme to create trust and reputation metrics in non-centralised open-membership systems.

2.3 Mobile payments

We now discuss mobile payment systems, also referred to as *mobile money*, that are primarily used in developing countries. These payment systems are relevant to our work on offline payments for financial inclusion, which we discuss in Chapters 5 and 6.

[19] See "Wells Fargo Bans Cryptocurrency Purchases on Its Credit Cards" `https://www.bloomberg.com/news/articles/2018-06-11/wells-fargo-bans-cryptocurrency-purchases-on-its-credit-cards`.

[20] We had mentioned this criticism in our Master's thesis. The final artefact of the Master's thesis, which was inspired by Laurie's papers and finalised in 2014 [17], was a distributed payment system in which supernodes process transactions and achieve consensus by replicating the ledger using a Byzantine Fault Tolerance (BFT) voting protocol (we used Practical BFT (PBFT) [32]).

The type of mobile payment systems we focus on are different from payment applications in developed countries, such as Apple Pay or Android Pay which are in effect extensions of EMV ported to smartphones: they connect users to banks and the traditional financial system. A major differentiator between mobile payments and EMV extensions is that the former usually offer their services primarily through the Subscriber Identity Module (SIM) that Mobile Network Operators (MNOs) provide to their customers (these systems are typically operated by MNOs), while the latter require both smartphones and conventional bank accounts, both of which are less widely used in developing countries. Since the MNO is able to program the SIM, another differentiator between mobile payment systems and payment apps is that the program and code for the former resides in the SIM. Moreover, these mobile payment systems are not directly connected to banks and the traditional financial system; the MNO (or another payment service provider) controls the customers' balances and performs the required operations to debit or credit accounts and finalise transactions.

2.3.1 Mobile payments overview

Mobile payment systems in Less Developed Countries (LDCs) rely on Short Message Service (SMS) messages or Unstructured Supplementary Service Data (USSD) sessions, supported by a SIM card issued by a phone company. If the payment operator is not an MNO, they can pay for access to a USSD channel operated by an MNO to provide their payment services. Thus, mobile payment systems require minimal connectivity in the form of GSM data transmission to communicate with the payment operator; cellular data plans connecting a device to the operator using 2G, 3G, or 4G are not required to process transactions. Therefore, mobile payments are accessible on all phones that implement the GSM standard. This low barrier to entry is crucial to provide services to users in LDCs who use "feature" phones (also called basic phones), which have minimal communication capabilities compared to smartphones; feature phones may not include a camera, Bluetooth or NFC connectivity, or any other form of ad-hoc communication such as Wi-Fi pairing[21]. GSMA Intelligence estimates that by 2020, the penetration of smartphones will be 57% in Africa and 66% globally (the estimate was 23% for Africa and 45% globally in 2015 when we started our research) [65]. Thus, payment operators who aim to attract many customers in LDCs must provide services that are compatible with feature phones. We focused our research to cater for users with feature phones, because our project was aimed at assisting financial inclusion efforts and the poorest demographics still widely use feature phones.

About 200 mobile-phone payment systems have been set up in various LDCs, and about 20 of them have become mainstream[22]. The "killer app" was migrant remittances in an environment where most people had no access to conventional banking services but where mobile phones were spreading rapidly. The best-known is Kenya's M-Pesa [64, 86] whose operator Safaricom processes payments amounting to a substantial proportion of Kenya's GDP[23], becoming a larger payment service provider than any local bank, and has become the monopoly MNO.

[21]It is not unusual that even if a device includes, e.g., a camera that it would be damaged and unusable, and must be supported with tape.

[22]See GSMA's "Mobile for Development" website (using the "Mobile Money" filter): `http://www.gsma.com/mobilefordevelopment/tracker`

[23]See mobile payment statistics at the Central Bank of Kenya: `https://www.centralbank.go.ke`

2.3.1.1 Payment operations and services

The SIM can verify a user PIN and contains keys for authenticating the customer; it can thus establish secure communication with a payment server. The server keeps customers' transaction histories, just like in a conventional bank. Customers can check their balance and make payments using menu options on the phone's screen, displayed by the trusted SIM card (the program residing on the SIM is called an *applet*). The phone number doubles as a bank account number. Cash-in and cash-out services are provided to customers by the payment operator to facilitate exchanging fiat currency into electronic float and vice versa. These services are operated by a network of mobile money *agents*, who earn commissions based on the services provided (e.g., withdrawing funds from the system incurs a fee).

To increase their own balance, a customer provides cash to an agent who will perform a transaction through the payment processor. This will result in the customer's electronic balance being increased by the amount they provided to the agent minus any transaction fees incurred (the agent earns a commission from the payment processor). The customer and agent are both notified of the result of the transaction. The customer can also verify their own balance at any time using their phone: the phone sends a balance-inquiry request to the payment processor. Some providers charge for inquiries and other services to rate-limit their use. To make a payment, a customer first enters the payee's phone number (shops have numbers prominently displayed, with large shops having short codes). The customer must then enter the amount, followed by their PIN to authorise the transaction. A payment message is sent to the server; if the funds are available, they are transferred, and the merchant or other payee is informed.

2.3.1.2 Reasons for success and remaining challenges

Attempts over the past ten years to set up similar phone payment systems in developed countries (such as Pingit and PayM in Britain) have generated little traction, as people in such countries already have plenty ways of paying both online and offline. There is no need to use an MNO's service in an already competitive financial system. In LDCs, however, banking services can be unreliable or absent, forcing individuals to look for alternatives: M-Pesa started as a micro lending service, but customers used it to pay each other, since banks were shut down for a few days when M-Pesa was being rolled out. Customers saw the MNO's service as a more reliable alternative with a lower barrier for entry: to open a bank account, customers need to provide proof of identity that many did not have and could not obtain, especially in the poorest communities – which did not have bank branches anyway. The MNO, through M-Pesa, provided an attractive and immediate alternative allowing customers to exchange cash for an electronic balance that they can then transfer to each other. A customer only needed to have a SIM provided by that MNO – and the MNOs had already built out networks of tens of thousands of agents nationwide. Later regulation for the mobile payment ecosystem had stricter requirements[24] for KYC and AML [2, 86]; that came only after the MNO solidified its control over the mobile payment market, making it difficult for newcomers to undermine their monopoly.

Regulation can be used after a monopoly establishes its presence to thwart attempts by newcomers to challenge it: the monopoly (Safaricom) uses regulation to accuse innovators

[24]We had to provide an official government ID to enable mobile payment services on our SIM when we visited Kenya to conduct our research.

of undermining its service, stalling the innovators' efforts in lengthy legal battles[25].

Some of the reasons mobile payments were adopted quickly are shared with other forms of electronic payments. In the context of developing countries, remittances are a major use case for mobile payments: workers in the city send their savings to their families in their home village. In contrast to using and carrying cash, mobile payments are generally considered safer. The traditional methods used to send remittances included the post, and taxi and bus companies: the migrant worker would mail banknotes, or give them to a driver travelling to their village to hand over to their family in return for a fee; in both cases, theft or loss are known risks. Moreover, the availability of the mobile payment service through a customer's phone, which can be accessed at any time, was a further reason for users to convert their money to electronic float.

However, even with a high penetration of mobile usage and mobile payment subscriptions, cash is still pervasive even in countries that pioneered mobile payments [73]. The most likely reasons for this, which we confirmed during our visit to Kenya, are transaction fees and unreliable network coverage that renders the payment service unusable [2, 98]. The Safaricom network is claimed to cover 91% of the economy, but it only covers 80% of residential addresses – excluding many of the poorest demographics. As an example of the unreliability of the service, Safaricom's M-Pesa was unavailable for 24-hour periods or more on various occasions [2, 120]. There is also congestion from time to time in Nairobi, which can leave people waiting at supermarket checkouts for 20 seconds awaiting payment confirmation. Another outage happened when MTN's mobile payment services in Uganda were inaccessible for about two weeks, frustrating its users, including the agents whose wellbeing and source of income rely on providing services to earn commissions[26]. Customers were left helpless and could not spend their electronic funds. This is especially concerning for the poorest demographics when they need immediate and reliable access to their funds. The issues highlighted here (fees and unreliable networks) are some of the main issues we try to address in Chapters 5 and 6.

We summarise in the following sections the strengths and weaknesses of mobile payment systems, as mentioned in this section and as relevant to our research.

2.3.2 Strengths

1. **Dedicated and tamper-resistant hardware**: As with EMV, mobile payments based on SIMs operate in a tamper-resistant environment, where all the critical operations are created within the SIM. Thus, the integrity of transactions is preserved.

2. **Low barrier to entry**: Mobile payments are accessible using any phone. No cellular data plan is required to enable most payment services. This substantially lowers the barrier to entry in order to make or receive payments.

3. **Usability**: Mobile payments use existing technologies, the mobile device and SIMs, without introducing unfamiliar hardware to the user. The payment process is as simple as sending a text.

[25]For example, see Safaricom's battles with Equity bank over the bank's use of overlay SIMs to circumvent the restrictions on phones (`https://www.standardmedia.co.ke/business/article/2000135850/safaricom-loses-battle-to-block-equity-bank-s-thin-sim-card`), and with BitPesa (`https://www.bitpesa.co/blog/bitpesa-v-safaricom/`).

[26]This case was included in an unpublished report by Microsave "Mobile Money for the Poor (MM4P) – The Customer Journey/Experience Research Report" which was provided to us by Microsave.

4. **Safety**: Using a mobile payment service from a safe location to send funds remotely reduces the risk of theft or loss. These risks are non-negligible when an individual has to transform the funds personally, or pay unreliable taxi or bus companies to transport cash for a fee.

5. **Availability**: Mobile payments, when the system is online and running, are available 24/7, in contrast to bank offices which close after normal business hours.

2.3.3 Weaknesses

1. **Unreliable network coverage and service disruption**: When network coverage is unavailable, it locks up the customers' funds and renders the service unusable. This can be a critical factor for adoption: if customers suffer a serious outage, they may cash-out their funds as soon as service is restored and never use the service again[27]. Just as with Bitcoin's "lack of offline capabilities" weakness discussed in Section 2.2.5, mobile payments cannot be processed when the network is unavailable, as there are no means for processing payments between peers directly.

2. **Transaction fees**: Incurring a fee for every transaction is inconvenient and costly especially when considering face-to-face and low-value transactions. As discussed in many reports [86, 2, 73, 49], this is of particular concern for financial inclusion efforts aimed at the poorest communities.

2.4 Micropayments

First-generation micropayments were proposed in the mid 1990s. We reviewed multiple proposals that suggest using micropayments to solve the "problem" of charging small amounts for Internet content (e.g., pay a penny for a single news article instead of a pound for the entire print edition). This included prominent designs such as PayWord and MicroMint [104], as well as NetCard [9]. Other designs aimed to solve scalability and overhead problems by introducing probabilistic schemes that do not result in incurring fees per transaction [102, 59, 70, 122]. Efficiency enhancements include outsourcing the acceptance of transactions to merchants instead of the bank (centralised provider), by creating transactions tied to the identity of the merchant as proposed in Millicent [59]. Decentralised verification alleviates the need to contact a bank to verify every transaction. The overview is based on simplified versions of PayWord, NetCard and Millicent.

2.4.1 Micropayments overview

Customer C has an account with bank B, and wants to pay merchant M. Using a credit-based scheme, B creates and signs[28] a *stick*[29] of values $(v_1, v_2, ..., v_n)^C_M$. This stick

[27] This happened after the mobile network in Uganda was shut down for several days during an election.

[28] In the simplified protocol shown, all messages are assumed signed, and the necessary freshness indicators such as nonces and timestamps are removed for simplicity. Micropayment schemes assume a Public-Key Infrastructure (PKI) exists to authenticate public keys of the principals involved in the payments and the bank; we later list this assumption and requirements as weaknesses.

[29] "Stick of coins" – banks handle coins in sticks that have perhaps 50 or 100 coins rolled up in a paper or plastic wrapper.

must include the identities of both C and M as payer and payee respectively. To create the stick of digests, B creates a chain of hash digests where $h_i = \mathsf{H}(v_i)$; H is a hash digest function (e.g., $\mathsf{SHA256}$), h_1 is the root of the chain, and v_i is the value to be hashed to create the digest; the result is a stick of hash digests $\big(\mathsf{H}(v_1), \mathsf{H}(v_2), \ldots \mathsf{H}(v_n)\big)^C_M$. B now sends the stick of hash digests to M, and sends the stick of values to C: this effectively opens a line of credit for C with M. C can now pay M by providing the values to M, where a payment is the pair (v_ℓ, ℓ); ℓ is the value of the current transaction added to the number of previously used stick values ρ (ρ is 0 initially). M can then verify that the value received from C generates the same hash value which is contained in the stick of hash digests received from B, by performing a lookup and a single hash digest operation then a result comparison. Then C has $n - \ell$ credit remaining. M can immediately verify that a transaction is valid, and can later send the highest payment (v_μ, μ) to B to close the line of credit:

$$
\begin{array}{rll}
\text{Request:} & & \\
C & \to B: & \text{“I want to pay } M\text{”} \\
B & \to C: & (v_1, v_2, \ldots, v_n)^C_M \\
B & \to M: & \big(\mathsf{H}(v_1), \mathsf{H}(v_2), \ldots \mathsf{H}(v_n)\big)^C_M \\
\text{Payment:} & & \\
C & \to M: & (\text{client's } v_\ell, \ell) \\
M & : & \mathsf{H}(v_\ell) =?= \mathsf{H}(\text{client's } v_\ell) \\
\text{Claim payments:} & & \\
M & \to B: & (v_\mu, \mu)
\end{array}
$$

This *offline* verification scheme does not require the involvement of B in each payment (besides bootstrapping the trust and line of credit initially), since the transaction can be verified locally, and the stick is merchant-specific; C cannot double spend credit.

2.4.2 Strengths

Micropayments use lightweight cryptography that can be quickly verified, along with decentralised verification to avoid points of failure:

1. **Lightweight cryptography**: Micropayment schemes use simple hash digests that can be computed quickly on modest CPUs. This is particularly important if the verification is performed on resource-constrained hardware.

2. **Decentralised, offline verification**: verification is decentralised and outsourced to the payee who can validate the payment and can prevent double-spend attempts (the stick is payee-specific). So the principals involved do not have to be connected to the bank, and can finalise the payment offline (a direct connection between a customer and merchant is sufficient to accept a payment.) This is in contrast to the cryptocurrencies discussed earlier where offline acceptance of a payment could open up the payee to a double-spend attack. Offline verification can produce efficient protocols as the bank does not need to be involved in verifying each transaction. If a decentralised verification approach is not possible, then low-value high-rate transactions could overburden the bank. Thus micropayment proposals illustrate how payee-specific lines of credit can enable offline verification, simultaneously increasing resilience and reducing server load.

2.4.3 Weaknesses

These micropayment schemes failed to gain adoption. The main problems with the early proposals can be summarised as follows:

1. **A solution looking for a problem**: Early proposals did not provide users with any added benefits. Charging users for separate articles rather than access to an entire collection was not attractive.

2. **Usability and unfamiliarity**: Micropayments also imposed an increased cognitive overhead. Users were not ready to change the way they buy things by worrying about multiple micropayments rather than a single larger payment [112, 113]; users did not care for being "nickel-and-dimed" [114].

3. **Lack of suitable infrastructure**: Key management issues are problematic. Vendors and brokers (using Payword's terminology; referring to merchants and banks, respectively) need to exchange public keys or validate certificates. Furthermore, smartcard tamper-resistance in the 1990s was not optimal to provide the necessary safeguards to protect critical data.

2.5 Summary of challenges and research goals

There are common problems with payment systems that we aim to address in our thesis:

1. **Resilience and scalability**: Most systems operate on fragile infrastructure that is vulnerable to DoS attacks and nation-state censorship. Yet payment systems normally work online, and fail to verify and accept transactions *offline* – that is, when communication to the payment system provider is unavailable for EMV-based systems and offline acceptance is not enabled; or, in Bitcoin, when a transaction cannot be sent to miners since online communication is unavailable. In Bitcoin, although the system is decentralised in its membership and participation, transaction processing is virtually centralised in a monolithic ledger; a more decentralised approach for processing transactions could increase system performance substantially. Micropayments may provide a useful means of outsourcing and decentralising payment verifications to the recipients, alleviating the need for always-on connectivity to a centralised verifier, and increasing system resilience while minimising verification overhead. These critical issues motivated our research. Therefore, we set out to provide mechanisms to alleviate the need to process every transaction online in order to achieve better system performance and resilience.

2. **Transaction fees**: In Bitcoin, transaction fees are used to demotivate spam, as well as prioritise a transaction to be processed faster (a higher transaction fee promotes a transaction to be processed faster when it is held in limbo in the Mempool). However, this means that during spam campaigns, transaction fees will rise so that valuable transactions can compete with spam transactions that otherwise take up space in Bitcoin blocks. This substantially increases the cost of non-malicious participation in the network, and the fees rise along with processing times, decreasing the appeal of Bitcoin as an alternative financial system (we discuss the details in Chapter 4). With regards to mobile payment systems, payment operators charge a

transaction fee for every transaction, harming the poorest demographics for whom transaction fees constitute a non-negligible share of their daily budgets. Payment operators justify transaction fees as necessary because 1) they provide a valuable service to their customers and this service adds overhead to their servers, and 2) that transaction fees rate-limit unwanted behaviour. Without fees, users can repeatedly send transactions back and forth overburdening the servers; this can scale into attacks if the process is automated resulting in high resource usage. We aim to offload the processing of transactions to users' devices in order to alleviate the burden of processing transactions centrally.

3. **Usability**: Usability can be a problem if new hardware and unfamiliar operations are introduced, creating mental overhead and decreasing adoption. Therefore, we aim to maintain acceptable usability levels, by mimicking existing systems with which users are familiar, and avoid introducing new hardware that is unfamiliar and can be costly to replace if lost or stolen. We aim to avoid increasing the cognitive overhead necessary to use the system, in order to increase adoption and minimise user errors.

4. **Lightweight protocols**: We utilise lightweight protocols that are simple to verify and require minimal resource usage. A motivation for using such protocols is the requirement to deploy offline payments on resource-constrained devices. Moreover, through lightweight protocols, we can produce cryptographic authenticators that are exchanged by users to finalise payments, while maintaining an acceptable level of usability.

The rest of this thesis is organised as follows. In Chapter 3 we study the social externalities of trust; how users can be incentivised to provide trustworthy nodes in a distributed system to help defeat service-denial attacks and nation-state censorship. This might be seen as a study on how we might crowdsource dependability. In Chapter 4 we consider how censorship and service-denial attacks might be used to undermine the Bitcoin ecosystem. This might be seen as a study of how a nation state might attempt to divide and defeat a payment system based on crowdsourcing. In Chapter 5 we take a more classic distributed-systems perspective on system resilience and scalability, by introducing a new way to process payments offline without relying on the availability of network coverage. Decentralising payment verification, and outsourcing the operations to the users, can alleviate the burden of contacting centralised systems to process every transaction. We also include the usability assessment of the prototype we built for offline payments in Chapter 6.

A secondary goal is to minimise the cost of providing payment systems, which then allows a payment provider to cut transaction fees. Moreover, by decentralising payment verifications that can be performed offline, we increase system resilience so users can continue making payments while a system provider is unavailable, and seamlessly maintain offline operations until a system is back online. We also use trusted hardware to address usability issues by minimising cognitive overhead and requirements from users to correctly handle critical data, and minimise the risks of data theft and tampering. Thus, we aim to produce an offline payment system that will work in less developed countries with acceptable usability levels and without introducing any unfamiliar hardware.

Chapter 3

The social externalities of trust

We first tackle an issue discussed in the Background chapter: that the lack of trust and reputation of nodes participating in decentralised systems exposes nodes to various attacks. These include overwhelming the network with connections to control the content relayed to those victim nodes, thereby undermining global communication (eclipse attack [62]), or hijacking connections to fragment the network into non-communicating quorums [12]. We investigate in this chapter how utilising a payment system, with a pseudo-currency, can create useful metrics for building trust and reputation, and incentivise correct behaviour.

Prior to working on this chapter, we reviewed micropayment proposals, as discussed in Section 2.4, since some of them are suggested as possible performance enhancements to reward participating nodes (which we review in the related work section in this chapter). We apply lessons learned from the Background chapter, such as decentralised verification discussed in Section 2.4.2.

The pseudo-currency is a proxy for trust and reputation metrics. These metrics add accountability to open-membership systems, such as the Tor anonymity network and Bitcoin. We think these metrics may be a necessary building block to add resilience to such systems.

The content of this chapter is adapted from our paper *Do you believe in Tinker Bell? The social externalities of trust*, by Khaled Baqer and Ross Anderson, which appears in the Proceedings of the International Workshop on Security Protocols 2015 [16].

3.1 Summary

In human societies, trust follows social consensus and crumbles when that is lost. However, the world of trust online is different. People trust CAs because they have to; Verisign and Comodo are dominant not because users trust them, but because merchants do. Two-sided market effects are bolstered by the hope that the large CAs are too big to fail. Proposed remedies from governments are little better; they declare themselves to be trusted and appoint favoured contractors as their bishops. Academics have proposed, for example in SPKI/SDSI [103], that trust should flow from individual users' decisions; but how can that be aggregated in incentive-compatible ways? The final part of the problem is that current CAs are not just powerful but all-powerful: a compromise can let a hostile actor not just take over your session or impersonate your bank, but "upgrade" the software on your computer. Omnipotent CAs with invisible failure modes are better seen as demons rather than as gods.

Inspired by Tinker Bell, we propose a new approach: a trust service whose power arises directly from the number of users who decide to rely on it. Its power is limited to the provision of a single service, and failures to deliver this service should fairly rapidly become evident. As a proof of concept, we present a privacy-preserving reputation system to enhance quality of service in Tor, or a similar proxy network, with built-in incentives for correct behaviour. Tokens enable a node to interact directly with other nodes and are regulated by a distributed authority. Reputation is directly proportional to the number of tokens a node accumulates. By using blind signatures, we prevent the authority learning which entity has which tokens, so it cannot compromise privacy. Tokens lose value exponentially over time; this negative interest rate discourages hoarding. We demotivate costly system operations using taxes. We propose this reputation system not just as a concrete mechanism for systems requiring robust and privacy-preserving reputation metrics, but also as a thought experiment in how to fix the security economics of emergent trust.

3.2 Introduction

Children know the story of Tinker Bell from JM Barrie's 1904 play "Peter Pan; or, the boy who wouldn't grow up". She is a fairy who is about to fade away and die, but is revived when the actors get the audience to declare their belief in her. The underlying idea goes back at least to ancient Greek mythology: the Greek gods' power waxed and waned depending on the number of men who sacrificed to them. More modern references include Jean Ray's 1943 novel *Malpertuis* [100] puts it as (translation from French): "Men are not born of the whim or will of the gods, on the contrary, gods owe their existence to the belief of men. Should this belief wither, the gods will die."

The idea that authority emerges by consensus and evaporates when the consensus does is not restricted to mythology; democratic institutions perform a similar function. In the context of a nation state, or even a professional society, they are developed into a governance framework optimised for a combination of stability, responsiveness and the maintenance of trust.

How are things online? The honest answer is "not good". When talking of trust online the first port of call is the Certification Authority (CA) infrastructure, which has many known failings. A typical machine trusts several hundred CAs, and trusts them for just about everything; if the Iranian secret police manage to hack Comodo, they can not only impersonate your bank, or take over your online banking session, they can also upgrade your software. Since an Iranian compromise caused the browser vendors to close down Diginotar, we have seen corporates moving their certificate business to the two largest players, Verisign and Comodo, in the belief that these firms are "too big to fail" (or perhaps "too interconnected to fail"). Firms hope that even if these CAs are hacked (as both have been), the browser vendors would never dare remove their root certificates because of the collateral damage it would cause. As for ordinary users, we trust Verisign not because we decided to, but because the merchants who operate websites we use decided to. This is a classic two-sided market failure.

Can we expect salvation from governments? Probably not any time soon. Governments have tried to assume divine powers of their own, first during the crypto wars by attempting to mandate that they have master keys for all trust services operating in their jurisdiction, and second by trying to control authentication services. Such initiatives tend to come

from the more secret parts of states rather than the most accountable parts.

Can we users ourselves do better? The SPKI/SDSI proposal from Ellison, Rivest and Lampson attracted some research effort in the late 1990s and showed how every individual user could act as their own trust anchor, but the question is how to deploy such a system and scale it up; the one user-based system actually deployed in the 1990s, PGP, remains widely used in niche applications such as CERTs and anti-virus researchers, but never scaled up to mass use. The application of encrypting email suffers from strong network externalities in that I need my counterparties to encrypt their email too. This has become the norm in specific communities but did not happen for the general population.

Can we scale up deployment in other applications from a club that provides a small initial user base? One case where this happens is in the Internet interconnection ecosystem, where trust among some 50,000 ASes is founded on the relationships between about a dozen Tier-1 providers, who form in effect a club; their chief engineers meet regularly at Nanog conferences and know each other well. But how could a service scale up from a few dozen users?

3.3 Motivation

In this chapter we present another example for discussion. We propose an anonymous online reputation system whose goal is to let people get better quality of service from a distributed proxy service such as Tor. Our proposed new trust service has limited scope; if it works, it can provide lower latency, while if it fails its failure should be evident. The more people trust it, the more effective it becomes; if people observe that it is not working and lose faith in it, then it will fade away and die. What's more, multiple such trust services can compete as overlays on the same network.

The Tor network [46] consists of volunteer relays mixing users' traffic to provide anonymity. The list of relays is disseminated through a consensus file which includes the IP addresses of all relays. IP addresses are required to allow a user's client (Tor software) to locally decide which relays to use to route traffic. However, an attacker (say a *censor*) can also download the consensus file, extract relay addresses, and block traffic by cooperating with local ISPs or using a nation-wide firewall. Thus, *victims* of a technically competent censor need private relays to connect to one of the publicly known relays. The private relay must not be known to the censor, or it too will be blocked. These private relays, called *bridges*, act as transient proxies helping victims to connect to the Tor network. Bridges are a scarce resource, yet play a critical role in connecting censorship victims to the Tor network. Therefore, we want to incentivise Tor users to run more bridges.

3.4 System design

Our proposed enhancement to Tor consists of competing *clubs*, each managed by a club *secretary*. This is a design difference from using a quorum of Directory Authorities (DAs) organised as a failover cluster, as currently implemented in Tor. The club secretaries, acting as Bridge Authorities (BAs), are responsible for disseminating information regarding club *members*. Each secretary is supported by a community of members who use its tokens as a currency to prioritise service. Members help censored users (*victims*) to circumvent censorship by volunteering to act as bridges, and can claim token rewards for their help.

To the secretaries, the performance of members is visible and measurable.

Secretaries clear each others' tokens, just as banks clear each others' notes. Through private token payments, we can analyse the behaviour of nodes and determine which are actively and correctly participating in the network. Tokens are blindly-signed objects used to request services from other nodes. Tokens lose value over time, to demotivate hoarding. We discuss now the details of operation.

3.4.1 Member registration

Members can join any club they choose; loyalty to a club is determined by the incentives and performance it offers. Members can participate in one club or many, volunteering for whoever provides reliable services. Members offer service to a club by broadcasting their services: "this is my key, address, etc and I'll be available for contact between 11:00 and 11:30; send victims my way". This process can be automated using an uncensored and trusted means of communication. We assume that most members are outside the censor's jurisdiction, though some will have ties of family or friendship to the censor's victims. Thus, some members may be motivated by the wish to help loved ones while others are altruistic and others are revenue maximisers. Some members will be within the censored jurisdiction.

We assume that keys can be exchanged successfully between members and secretaries: either the secretary publishes public keys somehow, or passes them on to new members as part of the recruitment process (about which we are agnostic). Within the censored jurisdiction, one or more designated *scouts* communicate with victims: we assume these are existing members. We assume the existence of innocuous store-and-forward communications channels such as email or chat; only a handful of censored jurisdictions ban Gmail and encrypted chat completely.

3.4.2 A simple threat model

Suppose that Alice and Bob belong to a club organised by Samantha to provide bridge services. Alice volunteers her IP address at time t to Samantha; a victim, Victor, contacts Sam to ask for a bridge; Sam gives him Alice's IP address and a short one-time password N_A; Victor contacts Alice and presents N_A; Alice shows N_A to Sam, who checks it, and Alice connects Victor to the Tor network. The protocol runs:

$$\begin{array}{lll} A & \rightarrow S: & IP \\ S & \rightarrow V: & IP, N_A \\ V & \rightarrow A: & N_A \\ A & \rightarrow S: & N_A \\ S & \rightarrow A: & OK \end{array}$$

The first problem with this simple protocol is that Samantha has to be online all the time; as she's a bottleneck, and the censor can take down the system by running a distributed denial of service attack on her.

Even without that we have two messages more in the protocol than we probably need. Our first attempt at improvement is to make the nonce N_A one that Alice can check; so let Alice share a key K_{AS} with Samantha and we construct the nonce N_A by encrypting a counter k with it. The protocol is now:

$$\begin{aligned} A &\rightarrow S: \quad IP \\ S &\rightarrow V: \quad IP, \{k\}_{K_{AS}} \\ V &\rightarrow A: \quad \{k\}_{K_{AS}} \end{aligned}$$

Alice can now check the nonce directly, so Samantha doesn't have to be online.

The next problem is harder; it is that the censor's shill, Vlad, can also ask for an IP address, and if Samantha gives him one, the censor will block it. This is the real attack right now on Tor bridges; the censors pretend to be victims, find the bridges and block them. Countermeasures range from restricting the number of IP addresses given to any inquirer, and trying to detect Sybil inquirers using analytics; but if you have a repressed population where one percent have been coopted into working for the secret police, then telling Vlad apart from Victor is hard (at least for Samantha who is sitting safely in New York).

3.4.3 A more realistic threat model

In what follows we assume that of the two representative club members, Alice is in the repressed country, while Bob is sitting safely in exile. Alice, if honest and competent, is better than average at telling Victor from Vlad, perhaps because of family ties, friends, or ethnic or religious affiliations. Alice might be undercover, or might have some form of immunity; she might be a diplomat, or religious official, or sports star. She might hand over bridge contact details to victims written on pieces of paper, or on private Twitter messages to fans. The full gamut of human communications, both online and offline, is available for members who act as scouts to get in touch with victims.

We now introduce another layer of indirection into the protocol. After Bob volunteers to be a bridge, Samantha gives the scout Alice a token for her to give to a victim Victor, constructed as $\{k, N_{AS}\}_{K_{BS}}$. When this is presented to Bob, he can decrypt it and recognise the counter, so he knows Samantha generated it for him, and grants bridge service to the victim. He sends it to Samantha, who can recognise it as having been generated for Alice, and can thus note that Alice managed to recruit Victor (or alternatively, if Bob's IP address then ended up on the blacklist, that Alice recruited Vlad by mistake). Formally:

$$\begin{aligned} B &\rightarrow S: \quad IP_B \\ S &\rightarrow A: \quad IP_B, \{k, N_{AS}\}_{K_{BS}} \\ A &\rightarrow V: \quad IP_B, \{k, N_{AS}\}_{K_{BS}} \\ V &\rightarrow B: \quad \{k, N_{AS}\}_{K_{BS}} \\ B &\rightarrow S: \quad N_{AS} \end{aligned}$$

N_{AS} might be constructed in turn by encrypting a counter; but once we start encrypting a block cipher output and a counter under a wider block cipher, we are starting to get to the usability limit of what can be done with groups of digits written on a piece of paper. As AES ciphertext plus an IP address is about 50 decimal digits. In some applications, this may be all that's possible. In others, we might assume that both scouts and victims can cut and paste short strings, so that digital coins and other public-key mechanisms can be used.

In a more general design, we have to think not just about running scouts to contact victims and tell victims apart from censors, but also about scouts who are eventually turned, and about clubs that fail because the club organiser is turned, or has their computer

hacked by the censor, or is just incompetent. We also have to think about dishonesty: about a bridge operator or scout who cheats by inflating his score by helping non-existent or Sybil victims. How far can we get with reasonably simple mechanisms?

3.4.4 Payment system

We avoid external payments, as offering cash payments as incentives to volunteers risks trashing the volunteer spirit (this is why the Tor project has always been reluctant to adopt any form of digital cash mechanism for service provision; volunteering is crucial for Tor's operation). Furthermore, we avoid using complicated zero-knowledge protocols or creating huge log files to protect against double-spending; large audit trails cannot scale very well. We prefer a lightweight mechanism that uses blind signatures made with regularly changing keys and member pseudonyms to provide privacy and unlinkability, as well as symmetric cryptography to create data blobs verifiable by the secretary or bridge. The token reward can then be used to pay for other services in Tor. For example, club members who run successful bridge services might enjoy better quality of service by using service tokens to get priority.

We now sketch a design using blind signatures rather than just shared-key mechanisms.

3.4.4.1 Member identifier

After registering a member, the secretary creates a series of data blobs as the member's *identifiers*. An identifier can be the result of encrypting the member's name plus a counter or random salt with a symmetric encryption algorithm and a key known to the secretary; identifiers change constantly, and the secretary alone can link them to members other than by context. As well as knowing which members correspond to which identifiers, the secretary also notes which victims are introduced to which identifiers as possible bridges. This is to make it hard for a member to generate fake victims and claim rewards without providing them with service. It does mean the secretary is completely trusted, but secretaries compete with each other to provide effective service. We discuss all this in more detail later.

3.4.4.2 Victim accounts

Secretaries maintain reputation scores not just for members but also for victims. Upon being introduced by a scout, a victim gets a default score of 1 allowing them to make a single bridge request. After a successful initial request, the victim's account is reduced to zero for the current period. This is one of a number of rate-limiting mechanisms to prevent Vlad from draining the IP pool.

Secretaries use victim accounts to mitigate a few possible attacks, which we explore in more detail in the discussion section. One purpose is Vlad detection: if the members a victim learns about are not censored after a period of time – the IP addresses are not blacklisted and the scout is not turned – the victim's account is increased; the opposite is true if a scout is arrested. Eventually, a diligent secretary should be able to identify fake victims by intersecting groups of victims and groups of censored members. It follows that the censor would have to refrain from blocking members if he wants to learn about new members. If he blocks members immediately, he betrays his shills. The conventional law-enforcement approach would be to block immediately, while the intelligence approach

would be to observe quietly until all or most of the members are identified. Forcing the censor to make a strategic choice opens up all sorts of possibilities.

Secretaries have an incentive to protect their members' identities; if the censor can spot and arrest the scouts and block the bridge IP addresses, the club will be ineffective and volunteers will help other clubs instead. Some volunteers will be picky, as if they join a badly-run club their IP address will be quickly blocked in that club's country of interest. Other volunteers may be happy to help victims in fifty countries and consider it a badge of honour to be blocked in a dozen of them. And if participation is rewarded not just with honour but with improved quality of service, then this will mostly come from jurisdictions in which volunteers are not blocked.

3.4.4.3 Token creation

Account payment mechanisms can be replaced by blind tokens at one or more stages in the process. In the simplest implementation, we can reward Bob for providing bridge services with tokens that offer better quality of service from Tor nodes. Given the way Tor works, this requires some form of anonymous payment or certification. So when Bob services a request from Victor, Bob can now use Alice's nonce N_{AS} to request an anonymous token from Samantha rather than just banking the credit. He does this by generating a well-formed token C_B, blinds it with a multiplier, and sends it to Samantha, who generates a blind signature [33] and returns it. With simple RSA blinding, where e is the public signature verification exponent, d the private signing exponent and n the public modulus, we have (the first four steps are carried from the protocol in Section 3.4.3):

$$\begin{array}{lll} B \rightarrow S: & IP_B & \\ S \rightarrow A: & IP_B, \{k, N_{AS}\}_{K_{BS}} & \\ A \rightarrow V: & IP_B, \{k, N_{AS}\}_{K_{BS}} & \\ V \rightarrow B: & \{k, N_{AS}\}_{K_{BS}} & \\ B \rightarrow S: & N_{AS}, r^e C_B & (\text{mod } n) \\ S \rightarrow B: & r C_B^d & (\text{mod } n) \end{array}$$

Bob now unblinds C_B by dividing out the blinding factor r. This token is unlinkable and can be used for interacting with Tor nodes. The token C_B includes a random number, generated by Bob, to detect double-spending. Unlinkable tokens can now be used to request other services by embedding them in the request; for example, if victims can handle public-key mechanisms, Victor might use such a token to request bridge service from Bob. However, it is in prioritising anonymous service requests that blind tokens really come into their own.

3.4.4.4 Defining time using key rotation

As noted earlier, we assume that new public signature-verification keys are announced for each future epoch. (It is possible to use an identity-based signature scheme by setting the public key for epoch i to be the value of i). We can simplify matters if we define a time interval as an epoch; this enables us to avoid using timestamps in token protocols (we'd prefer to avoid the complexity of using partially blinded signatures that contain timestamps). Changing public keys frequently also reduces the amount of state that must be retained to detect double spending, a known problem with blind payment systems. If tokens are used only by Tor nodes with high-quality network service, this is probably a

reasonable simplifying measure. We note in passing that nodes have an incentive to pay attention to the signature traffic, to ensure that a cheater does not pass them a stale token.

3.4.4.5 Validation and value of tokens

Tokens expire if they were issued using a signing key that was retired or revoked; a signing key may be deemed retired if it was first used a certain number of epochs ago. We propose that the number of epochs since the token's signing key was first used is a deflator, which will decrease the value of a token. The formula used might be exponential, to represent a negative rate of interest; it is not clear that it matters all that much whether deflation is exponential or linear. Club secretaries can refresh tokens with new ones of an appropriately lower value, according to rules we will discuss later, and will reject double spend attempts. Expired tokens will also be recognised and rejected by all nodes, limiting the volume of data needed to detect double-spending.

3.4.5 Generating trust and reputation metrics

Each secretary acts as a bank and keeps members' accounts: each member's balance is the amount of correctly performed identifiable service plus the number of tokens claimed or refreshed in each epoch. These balances act as a proxy for reputation. The balances also deflate over time, but significantly less quickly than tokens in circulation. For example, if a token loses 20% of its value at each epoch when in circulation, it might lose only 5% when banked. Thus a member wanting to maximise its reputation has to do useful work and bank tokens promptly. The history of members' bank balances may also be made available to other members; there are second-order issues here about the potential identifiability of members involved in particular campaigns, so whether the secretary publishes member account history or smoothed metrics derived from it would depend on the application.

The overall effect is that the network as a whole can take a view on how much it trusts particular members. The more tokens Bob has in the bank, the higher his reputation. Note that there is little incentive for a group of colluding nodes to manipulate the reputation system by trading among themselves; they achieve nothing except to decrease their original endowment of tokens by the amount of time these tokens are not in a bank.

Another advantage of using a rapidly depreciating currency is that we can use the reputation system itself as an indicator of member liveness. We want to avoid sending requests to offline nodes; yet if we contact each node directly to request proof of liveness (for example with an ICMP packet), we open up a denial-of-service attack where a malicious node saturates the system with liveness checks. A real-time reputation system may help us avoid this.

3.5 Discussion

The system proposed here is concerned with enabling a group of users to maintain situational awareness in a censorship avoidance system. Each club of users can set up their own bridge authority maintained by a club secretary; an authority trusted by more users will become larger. In equilibrium we hope that clubs would settle each others' tokens, just as banks clear each others' notes.

3.5.1 Mitigating collusions and malicious members

There are a number of possible ways in which members might behave improperly. Alice and Bob might collude, so that Bob pretends to service Sybil victims invented by Alice; Samantha can mitigate the risks of this to some extent by randomising the allocation of IP addresses to scouts, and running appropriate analytics. The worst case is if the target country's intelligence service manages to hack Samantha's computer; then it is game over. For that reason we want multiple competing clubs. We note that while some nation states have had multiple competing intelligence services, it is rarer to find multiple competing counter-intelligence organisations; perhaps our construction can find other uses.

The next most severe attack might be if the target country's intelligence service manages to subvert Alice and most of her fellow in-country members, and uses their bridge resources to make innocuous network connections, rather than blacklisting them, thereby denying the resources to censorship victims and denying both Bob and Samantha knowledge that Alice has been subverted[1]. While a normal censor will act as a policeman and block bridge IP addresses, a more strategic adversary might prefer to leave them be and exhaust the resource. Detecting and defeating such attacks requires further channels of information. Ultimately we rely once more on the facts that there are multiple clubs, and multiple channels of communication between censorship victims and their family, friends or co-religionists in exile.

3.5.2 Mitigating Sybil attacks

Members can claim tokens by fabricating victims, but those members end up "burning" their victims' account balances (and their email identities) if they don't use the identifiers. Recall that victims are assigned to members randomly, and Samantha can run analytics to determine which scouts and which bridges have outlying patterns of victims. Diagnosis is not always going to be straightforward; Samantha might suspect that some victims are bogus, or that some members refuse to service victims, but it may in fact be the case that some victim group cannot contact members due to censorship or DoS attacks.

It does little good if multiple members collude to exchange each others' identifiers; they end up burning their fabricated victims' identities, as long as there are honest members acting correctly and serving genuine victims, and the secretaries assign victims (clients) to members (bridges) randomly while monitoring performance and correct operations. This restricts the victims' ability to select members, which is similar to TorCoin, through its TorPath mechanism, where an "assignment server" assigns a Tor path to clients (we provide more detail in Section 3.6). Therefore, members are better off acting correctly.

3.5.3 Metrics for security economics

In this proposal, we attempt to incentivise correct behaviour, to generate metrics to identify which nodes are most interconnected, and empower nodes to shift trust to other nodes. In other words, we facilitate the mechanisms required to democratise trust and power, by empowering participating nodes to vote (transact using tokens) for the node that deserves their trust. Moreover, by creating a system to generate useful metrics, this

[1]Such subversion might involve a national-scale malware implementation programme; see for example Gamma's "Project Turkmenistan" disclosed on Wikileaks.

design can be used to facilitate research on the security economics of users' interactions, inspired by [1, 45].

3.6 Related work

In their paper *On the economics of anonymity* [1], the authors argue that the actions of interacting nodes in the network must be visible for informative decision-making about malicious behaviour. Through the trust and reputation metrics introduced in this chapter, we can now understand node interactions better while preserving privacy. A thorough and insightful discussion based on practical experience is provided in [45], whose authors state that in order to provide the mechanisms for verifiable transactions and reputation ratings in anonymity systems, they needed to retrofit appropriate metrics. In fact, those authors already wondered whether it might be possible to create a reputation currency that might "expire, or slowly lose value over time". The redesigns that the authors discuss in [45] were originally introduced in [43] and [41]. In [43], the authors integrate into Mix networks (anonymous remailers such as Mixmaster [88]) the role of witnesses: semi-trusted nodes that act as referees to service rejection or abnormal behaviour (but this introduces multiple trust bottlenecks and can be abused if the witnesses are compromised). In [41], the authors decided to drop the witness construction. Network paths are constructed in cascades; if one node in the path fails, every node in the path is rated negatively. Without proof-of-service-failure to pinpoint the node responsible, this design is vulnerable to an adversary joining multiple correctly-operating cascades, perhaps in order to route traffic to other adversary-owned cascades (which would de-anonymise users). Similarly, Free Haven [44] uses reputation to reward correctly-operating servers that store data and fulfil their contracts. The reward is a higher reputation that allows a server to store its own data with other servers (so long as no issues occur when validating service contracts). In fact, Free Haven is one of the first designs to use reputation as a form of currency. Moreover, the stamp-trading system discussed in [90] suggested that reputation built through proofs of providing services can be used as a currency to facilitate node interaction.

In the context of anonymity networks, there have been many proposals to rate and incentivise service-providing nodes (Tor relays). Most of these designs rely on bandwidth verification. For example, in [42], the authors suggest granting priority to the traffic of high-bandwidth nodes using gold stars. However, this can profile relays that also run clients by isolating the gold star traffic from the ordinary variety. More traditional approaches include XPay [35] and PAR [10] which use digital cash to reward relays. A more novel approach is discussed in BRAIDS[2] [67] which employs a similar architecture to the one proposed in this chapter. BRAIDS uses partially blind signatures to embed timestamps in tickets. This allows the Directory Authority (DA) in a BRAIDS-enabled system to expire tickets based on timestamps. In contrast, we expire tokens by key rotation, which enables tokens to be unlinkable. BRAIDS nodes create relay-bound tickets that can be verified by the relay; this increases efficiency, but limits the freedom to transact, as a relay has to contact a DA to issue new tickets for other relays. If a Tor path includes a relay that refuses service, the node must create new relay-bound tickets (generating another request to DA), since the original tickets will not be accepted by another relay. In LIRA [68], the authors attempt to increase efficiency by introducing a probabilistic

[2]We initially designed our system without knowledge of BRAIDS then amended this chapter to refer to it, but did not set out to design an improvement to BRAIDS.

micropayment protocol into their design (in fact, the authors use a similar construction to MicroMint [104]). In rBridge [121], the authors aimed to reward users using a credit system based on how long information about a Tor bridge remains secret from an adversary (measured by its reachability and non-censorship). This credit system can then be used to obtain information about another bridge if the original bridge is censored.

Other authors were inspired by cryptocurrencies: In [69], high-bandwidth relays are awarded with *Shallots* which are redeemable for *Priority Passes* that can be used to classify and prioritise Tor traffic. In TorCoin [58], TorPaths are used to verify paths constructed in Tor, and can be viewed as an enhancement for Tor's bandwidth verification. Another approach that aims to preserve users' privacy was proposed in [23] that uses Proof-of-Work shares as micropayments for relays. Relays can then submit those shares to a mining pool and claim rewards in bitcoins [95]. This provides anonymity for users but not for relays if they use the pure Bitcoin protocol.

Trust and reputation metrics are used for various reasons. For example, Advogato [78] was used to create attack-resistant metrics to correctly rate user-generated content. Another prominent example is Google's PageRank [99], which rates web pages based on how many other pages point to it and creates a reputation rating for how reliable a page is (essentially, the pointers to a page are votes for that particular page but are weighted recursively by their own reputation).

A common problem with most proposals for incentivising Tor relays by using bandwidth verification schemes, or user-generated feedback, is that the authors do not discuss Sybil attacks which involve the adversary creating many circuits to their own relays to game the system. We attempt to mitigate this through the random assignment of victims to members, while monitoring performance to observe correct behaviour (a task given to a trusted secretary in our system). This is similar to the trusted role of "assignment servers" used in TorCoin (as discussed above); a TTP is responsible for assigning tasks and monitoring performance to evaluate members.

A further issue is that an overlay on Tor that enables some club of users to enjoy priority service would risk a substantial decrease in the size of the anonymity set. In our proposal, a large number of clubs can each have a secretary acting as their own DA, and the secretaries can clear each others' tokens.

3.7 Conclusion

In this chapter, we sketched a design for an anonymous reputation system that can provide a quality-of-service overlay for an anonymity system like Tor or an open-membership system like Bitcoin [95]. Unlike most electronic trust services today, it has the right incentives for local and democratic trust management. Groups of users can each establish their own token currency to pay for forwarding services, and the nodes that work hardest can acquire the highest reputation, enabling them to get still more work. Groups can clear each others' tokens. And finally, any group that fails to compete will find that its failure becomes evident; its users can desert it for other groups and it can just fade away.

3.8 Afternote

After contacting lead developers of anonymity systems, we learned that the reservations they have for introducing payment and reward schemes to their systems stem mainly from non-technical reasons. The main concern is that introducing such schemes to purely voluntary systems would change the dynamics from nodes operating for the benefit of all others, to a predatory system that motivates adversaries to accumulate rewards while possibly producing damaging effects to these systems. Thus, we decided not to pursue the goal of deploying similar schemes to build trust and reputation metrics. However, we note that the system proposed in this chapter can alleviate network attacks discussed in Section 2.2.5, by ranking participating nodes and creating a reputation score for them in order to mitigate malicious activity.

Moreover, proposals incentivising Tor bridges include using cryptocurrencies which we discussed earlier (e.g., as proposed in [23] using schemes similar to those used by Bitcoin mining pools), to reward nodes. Therefore, we focused next on investigating performance issues in cryptocurrencies that create critical bottlenecks and detrimental effects that undermine system performance, scalability and resilience. We discuss the details in the next chapter.

Chapter 4

Bitcoin stress testing

We discuss in this chapter a few issues with Bitcoin's throughput and performance. We analyse a spam campaign that occurred in July 2015, which crippled system performance by overwhelming the network with spam transactions. This resulted in Bitcoin transactions experiencing more delays in processing time (reducing the reliability and efficacy of the payment system), and incurring higher transaction fees to prioritise transactions (increasing the cost of participation in the network). This shows that Bitcoin requires more research to add resilience against undesirable content being included in the blockchain, which take valuable space within blocks away from non-malicious transactions waiting in limbo to be processed and finalised.

The content of this chapter is adapted from our paper *Stressing out: Bitcoin "Stress Testing"*, by Khaled Baqer, Danny Yuxing Huang, Damon McCoy, and Nicholas Weaver, which appears in the Proceedings of the Third Workshop on Bitcoin and Blockchain Research (2016), affiliated with the International Conference on Financial Cryptography and Data Security [19].

4.1 Summary

We present an empirical study of a recent spam campaign (a "stress test") that resulted in a DoS attack on Bitcoin; its goal was to understand the methods spammers used and the impact on Bitcoin users. To this end, we used a clustering-based method to detect spam transactions. We then validated the clustering results and generated a conservative estimate that 385,256 (23.41%) out of 1,645,667 total transactions were spam during the 10-day period at the peak of the campaign. We show the impact as increasing non-spam transaction fees from \$0.11 to \$0.17 (per kilobyte of transaction) on average, and increasing delays in processing non-spam transactions from 0.33 to 2.67 hours on average. We estimate the direct cost of this spam attack in terms of increased mining fees at \$49,000; an attacker prepared to spend this much could DoS Bitcoin in 2015.

4.2 Introduction

The Bitcoin network [95] was subjected to a major spam campaign during the summer of 2015 that caused degraded performance. The likely intent of the incident (advertised as a "stress test") was to flood Bitcoin with spam transactions, in order to expose its vulnerability to spam and DoS attacks, and to garner support for a proposed change to

increase the number of transactions that the Bitcoin network can verify, which was then approximately 3 transactions per second. DoS attacks against Bitcoin had been theorised. However, to date there has been little empirical analysis of DoS attacks launched directly against Bitcoin.

4.3 Background

We already gave a high-level description of the Bitcoin protocol in Chapter 2. The main components of a transaction, relevant to our analysis here, are the transaction ID (*txid*), the inputs to the transaction (*vin*), and the outputs (*vout*). A transaction includes inputs that reference one or more unspent transaction outputs (UTXOs). Transactions are broadcast to peers in the Bitcoin P2P network, and the transaction propagates through the entire network in about 13 seconds [39]. Received transactions are checked and maintained in a node's own local memory pool (*Mempool*), where they remain in limbo until confirmed and included in a block. Although a node may maintain unconfirmed transactions for a long period of time, memory pressure may cause a node to evict older entries from the Mempool if it grows sufficiently large.

Nodes also maintain an unspent transaction output set to easily verify inputs to newly received transactions. Any increase in the UTXO set adds memory pressure on nodes which hold the UTXO set in RAM. Unlike with the Mempool, this pressure cannot be relieved by eviction, but requires changing the node's implementation.

4.3.1 Transaction priority

In the reference implementation, a Bitcoin miner calculates a transaction priority P and uses this to determine which transactions to include in the block. To calculate P, the node considers all inputs to the transaction as well as its size. P is defined in Bitcoin as $\sum_{i=0}^{n}(value_i \times age_i) \div S$, where n is the number of inputs to the transaction, *value* is the value of input i (in Satoshis[1]), *age* is defined as the difference between the current block's height and the input's block height, and S is the transaction's size. The value of P determines a transaction's fate, and there are three possibilities:

1. Include transactions in the high-priority section of a block (50 KB); no transaction fee is necessary. The following conditions must be satisfied, the transaction must be:
   - smaller than 1 KB
   - all output values are at least 0.01 BTC
   - P is high as determined by $value_i$ and age_i
2. Transactions that pay fees are prioritised by highest mBTC per KB.
3. The remaining transactions are maintained in the Mempool until one of the two conditions above is satisfied.

In the latter case, *age* is the determining factor for P since everything else is constant. It is of particular note that miners prioritise for higher fees.

[1] 1 Satoshi = 10^{-8} BTC.

4.3.2 DoS targets inherent in Bitcoin

Spam can be detrimental to the Bitcoin network by outcompeting legitimate transactions for inclusion in a block, delaying other transactions. We define the following types of spam:

1. **Fan-out**: Transactions that split a few inputs into many outputs occupy space in the blocks and also increase the UTXO set.

2. **Fan-in**: Transactions which absorb a large number of inputs reduce the UTXO set but still occupy substantial space in the blocks.

3. **Dust output**: Transactions that create very small "dust" outputs convey a trivially small amount of value but occupy the same amount of resources in the network.

The spam campaigns in the "stress test" target one or more aspects of the Bitcoin environment, including the block size limit, the UTXO set, and the computational cost of verification. All these limited resources represent potential targets.

The primary publicly-stated motivation behind the stress test campaign was to provide a justification for raising the Bitcoin block size limit before organic demand limits the ability of Bitcoin to process payments. The current Bitcoin block size of 1 MB globally supports less than 3 Bitcoin transactions per second. Since this is three orders of magnitude lower than Visa's sustained rate of 150M transactions per day (and peak processing ability of 24,000 transactions per second) [115], it was argued that Bitcoin payment processing was insufficient to meet the ambitions of the Bitcoin community. The spammers claimed that they wanted to demonstrate the impact of this limit by squeezing out normal transactions.

Raising the block size, however, opens up a different DoS vulnerability: a long-term growth DoS on the Blockchain itself. Since the Blockchain records all previous transactions, an attacker could perform low-fee transactions simply to take up space. Thus if Bitcoin raised the block limit to 20 MB, and an attacker can cheaply consume 10 MB of data per block, this would cause the Blockchain to increase in size by half a terabyte a year.

Since valid transactions can only spend unspent outputs, most full Bitcoin nodes keep the UTXO set in memory to speed validation. The memory requirements for the UTXO set are solely based on the number of unspent outputs, so the inclusion of dust outputs in the stress test adds memory pressure to the UTXO set. A better-designed Bitcoin node should not have this vulnerability.

Another DoS attack occurred on October 7 and 8, which also put a significant amount of pressure on the Mempool memory, raising the Mempool to nearly a GB, with a transaction backlog of nearly a week. Since a large number of nodes run on Raspberry Pi devices and other constrained systems, this large Mempool managed to crash over 10% of all Bitcoin nodes[2]. Most of the spam itself, however, was of low priority. Such spam does not put pressure on block inclusion, but neither does it cost the spammer any bitcoins; transactions that are never confirmed do not incur a cost for the sender.

An inadvertent CPU DoS occurred due to a mining-pool's "cleanup" block, a single 1 MB transaction that served to remove a massive number of unspent transactions sent to crackable "Brain wallet" addresses (which use a passphrase, instead of private keys, to create Bitcoin addresses and spend bitcoins). Other nodes required substantial CPU time

[2] `https://www.reddit.com/r/Bitcoin/comments/3ny3tw/with_a_1gb_mempool_1000_nodes_are_now_down`

to validate this block, as the current implementation required $O(n^2)$ time to validate a transaction. There may be other CPU DoS possibilities inherent in the Bitcoin protocol that attackers can exploit.

Another DoS is inherent in "transaction malleability". Someone can take a valid transaction, permute it so it has a different *txid*, and broadcast that modified transaction to the network. If the attacker's transaction is accepted into the blockchain, this can disrupt wallet services, hardware wallets, and other systems tracking *txid*s to determine when a transaction commits to the blockchain. Recently, an attacker performed this DoS "because I am able to do it."[3]

Finally, a later (failed) spam campaign attempted to flood the network with invalid transactions, perhaps intending either a traffic DoS or a CPU DoS. The "money drop", a public release of private keys by one of the purported instigators of the stress test, seems intended to cause a big race which would cause a large number of "double-spend" transactions. This did not produce a meaningful disruption of the network, although it was probably intended to introduce computational load.

One aspect not encountered during the stress test was the effect of filtering valid but spammy transactions. The introduction of spam filters, if an unknown attacker continued a longer-term DoS attempt, could in itself be a DoS. If the attacker adapts to the filters, eventually the filters will either fail to stop the spam or incur false positives. Even a small false positive rate might be disruptive: could a payment network tolerate a 1-2% transaction failure rate due to spam filters?

4.4 Data collection

In our study, we set up a server connected to a public-facing network. We installed Bitcoin Core 0.11 and kept it running between June 19 and September 23, 2015. We collected three main data sets using Bitcoin daemon's JSON-RPC interface.

1. **Bitcoin Blockchain**: On September 23, we downloaded the entire blockchain using the `getblock` and `getrawtransaction` methods. This returned details for all blocks and transactions, such as the timestamps of blocks, the timestamps at which we received the transactions, the number of transaction inputs and outputs, as well as the input and output amount. We stored the data as plain-text JSON strings. As a result, the total data size is 350 GB.

2. **Mempool**: Between June 19 and September 23, the `getrawMempool` method was invoked every minute. This returned a list of unconfirmed *txid*s currently in the Mempool. These would be either committed to the blockchain or later discarded by the P2P network. We saved this list of *txid*s, along with the timestamp of the RPC call, on the Hadoop file system. During this period, we captured 12 million distinct *txid*s in the Mempool, amounting to 250 GB of plain-text data.

3. **Unconfirmed transactions**: For every unconfirmed transaction that we obtained, we immediately looked up the transaction details using the `getrawtransaction` method, since the Mempool could discard the transaction at any moment. To optimise for speed and storage, we ignored transactions we had previously seen.

[3] `https://bitcointalk.org/index.php?topic=1198032.msg12579271`

Table 4.1: Data sets. All data sets cover a period between June 19 and September 23.

Data	Period	Size
Blockchain	Between Jan 9, 2009 and Sept 23, 2015	350 GB
Memory pool	Between June 19 and Sept 23, 2015	250 GB
Unconfirmed transactions	Between June 19 and Sept 23, 2015	1.3 TB

Table 4.2: Transaction features

Feature	Notation	Description
Inputs	I	Number of inputs
Outputs	O	Number of outputs
Ratio	R	$I \div O$
Priority	P	Value-weighted measurement
Size	S	Size (bytes)
Size and Ratio	$S \times R$	Emphasise fan-in and fan-out
Fees	F	Value of unclaimed outputs
Coin days destroyed	CDD	Coin age and spending velocity
Value	V	Total output value
Fees to values ratio	$F \div V$	Emphasise fee differences

Finally, we saved all the transaction details, along with the data collection timestamp, on Hadoop. Between June 19 and September 23, we captured 1.3 TB of unconfirmed transactions in plain text.

The total volume of data collected is 2 TB, which we saved as plain-text JSON strings on the Hadoop file system and analysed with Spark. We summarise our data sets in Table 4.1.

As we collected data using only a single node, our perspective of the P2P network, and thus the transactions in the Mempool, is potentially biased. In particular, network propagation takes time. For transactions in the Mempool, the timestamps that we observed may be later than the originating timestamps. Furthermore, whether a transaction is relayed is up to individual nodes. A transaction created a few hops away is not guaranteed to reach our node. It is, however, beyond the scope of this research to adjust for such biases. We assume that our observation of the network is largely consistent with the rest of the network.

4.5 Spam clustering

We use an unsupervised machine learning method, k-means clustering, to find similarities and evaluate our findings. This is not necessarily a perfect filter, but, as we manually verify, this does efficiently detect the spam transactions in the "stress test".

To use k-means clustering, we create a multi-dimensional vector representing features of a Bitcoin transaction. We include in Table 4.2 the list of features and follow up with defining features that were not previously discussed.

R is necessary to highlight the difference between fan-in and fan-out transactions. We further highlight this difference by multiplying the size of the transaction by its

ratio (otherwise, transactions with clear differences in R are clustered together based on similarities in S). We include another property to highlight the velocity of spending bitcoins represented as CDD[4]. This feature gives more weight to older coins, and can be calculated as $\sum_{i=0}^{n}(value_i \times age_i)$. Unlike P, CDD does not consider S, *age* is measured in number of days rather than blocks (an estimate of 144 blocks are produced each day), and *value* is in bitcoins.

4.5.1 Methodology

Since spam campaigns may not link transactions and addresses, parsing the blockchain to look for linked transactions might be a futile process. Our approach is different: we cluster transactions based on their motifs (trends in the Bitcoin network), and disregard transactions' identifying information (output addresses, *txid*, etc.). Our main assumptions at this stage echo those required for machine learning algorithms: a pattern exists, we cannot mathematically point out differences in patterns (without data visibility), and we have a large trove of data to show the patterns exist. We assume motifs do exist because spam requires construction in-bulk to have a measurable effect on the network. Thus spammers naturally create large numbers of transactions that "look similar". We also expect that such groups of transactions may have different motifs compared with normal Bitcoin behaviour, since spammers want to minimise the cost and maximise the impact, producing different types of transactions (e.g., very high fan-out or dust output) that particularly stress the network.

What we seek is a high-level interpretation of the data into distinct clusters that we can then use to label transactions as spam and validate our results. Thus, to investigate our main goal of identifying spam motifs, we consider the entire Bitcoin network as an entity, rather than analysing features of a transaction independently from network norms. The latter process relies heavily on what features should be considered to identify spam, which might assign more weight to some features while disregarding others that are more influential.

We use k-means clustering, as provided in Spark's machine learning library (`MLlib`). k-means clustering is a type of machine learning algorithm for unsupervised learning. This algorithm is particularly useful to cluster similar data together when it is non-trivial to define *similarity* using the unlabelled data. Similarity of vectorised data is determined using k-means by minimising the Within-Cluster Sum of Squares (WCSS); the data is matched to the cluster centroid with the closest mean. The following equation is used to iterate over the data to get optimal cluster centroids in order to minimise WCSS: $min \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$, where k is the number of clusters, x is the data element (in vector form), S_i is the set containing n elements, and μ_i is the mean of S_i (i.e., the mean of all the elements in vector form that are contained in S_i).

To reproduce the results discussed in this chapter, the following properties of k-means must be considered: the number of clusters k was set to 10, the number of `maxIterations` was set to 100, and `initializationMode` was set to `random`. The silhouette coefficient measures the homogeneity of the data in a cluster. This is performed by measuring the average dissimilarity (defined in terms of distance between data elements) between a given element within its cluster, and comparing the result with the average dissimilarity between that same element and elements of another cluster considered to be the next

[4]This feature is used by Bitcoin block explorers, see for example: `https://blockr.io`

best-fit. However, in our case, our aim is to show general transaction motifs, rather than to show detailed transaction differences or find anomalies. We arrive at $k = 10$ after testing multiple values for k to show enough visibility of transaction patterns. If we choose $k = 11$ for example, we obtain a new cluster where the average of transaction outputs is 8 rather than 11 (as shown in cluster 9 in Table 4.3). Instead, we accept that the clustering algorithm groups these transactions together in cluster 9, given that they are similar in other features. Conversely, with $k < 10$, clusters contain transactions that differ in most of their features; this does not enable us to inspect the clusters to easily determine which of them fit our definitions of spam. With $k = 10$, we see the "outliers" visible in a dedicated cluster (cluster 8 in Table 4.3), whereas with $k < 10$ these outliers are included in other clusters that do not match well.

The initial step for processing data was weeding out some transactions that alter the clustering results. To set a starting point, we create two checks to filter transactions. First, we check if the transaction creates dust output (we explain this check in detail later). The second check determines if the transaction's fan-out ratio is unusual (a threshold is set at 0.3). The rationale for these two checks is as follows: If a fan-in transaction creates dust output, then it qualifies as spam, otherwise it is minimising the set of UTXOs that must be maintained to verify transactions. Moreover, if a fan-out is unusual, this is enough to qualify a transaction for clustering, and we later determine if the transaction is spam by inspecting clustering results, and checking for dust outputs in clusters that seem to contain normal transactions.

We analyse confirmed transactions that occurred between June 24 and July 17, 2015. The total number of transactions in this epoch is 3,321,429. To obtain k-means clusters, we perform k-means training on all transactions that were confirmed during the July spam campaign epoch, that occurred between July 7 and 17, the total number of transactions in this training epoch is 1,645,667. Using the cluster centroids from the spam epoch, we analyse the pre-spam epoch to validate our results.

4.5.2 Results and motifs

We now discuss motifs found in more than 1.6M transactions that occurred during the spam epoch. Table 4.3 shows each cluster centroid's features. As discussed earlier, these centroids are the result of optimising WCSS, and are represented as the means of the values of all transactions in the corresponding cluster. Table 4.4 shows the standard deviation of the cluster centroids[5].

1. **Fan-in. Clusters 2 and 4** include about 14K fan-in transactions. The pattern is distinct: large I and one O (in rare cases O is for two addresses). The transactions vary in S due to variations in I, and a notable distinction is in *CDD*. Cluster 2 includes larger values for *CDD*, which indicates that the inputs are not used for rapid transfer of value. Moreover, these transactions may not have been used as spam *per se*, but are rather part of tumblers or mixers where a large number of inputs are collated into single outputs and the chain continues, in order to mix coins together and obtain relatively better privacy. These transactions involve long chains of many inputs to a single address, the last address then transfers funds to multiple outputs

[5]The notation used in the tables corresponds to the notation used for the transaction features defined earlier. Note that both tables include rounded values, while attempting to maintain distinctions for small values with the minimum amount of rounding necessary. For better presentation, we omit some features.

Table 4.3: Cluster centroids (*confirmed transactions*)

C	*TXs*	*I*	*O*	*R*	*P*	*S*	*F*	*CDD*	*V*
0	48K	1.35	46	0.06	0.74	1.8K	0.0004	0.195	4.06
1	28	4.4K	1	4.4K	0.001	645K	0.04	0.06	0.0
2	896	106	1	103	0.17	16K	0.001	0.34	0.13
3	20	1.1K	1	1.1K	0.0008	162K	0.01	0.012	0.0
4	13.5K	31	1	31	0.04	4.7K	0.0002	0.02	0.006
5	16	1.4	13	0.15	535K	668	0.0004	25K	1K
6	9.5K	20	17	19	0.4	3.5K	0.0004	0.14	1.4
7	425K	1.1	2	0.8	1	224	0.0001	0.022	1.43
8	2	1	19	0.05	136M	787	0.0002	740K	3K
9	117K	1.2	11	0.14	72.43	561	0.0002	2.7	6.5

Table 4.4: Standard deviation of selected features (*confirmed transactions*)

C	*I*	*O*	*R*	*P*	*S*	*F*	*CDD*	*V*
0	4	104	0.77	27	3.6	0.002	17	40
1	1.2K	0	1.2K	0	176	0.012	0.05	0
2	43	0.2	35	2	6	0.0005	4	1.8
3	403	0	403	0	60	0.004	0.02	0
4	8	0.1	8	0.8	1.2	0.0002	0.5	0.24
5	1	7	0.1	0.35M	0.38	0.0001	26K	1.2K
6	2	0.4	2	1.65	0.35	0.0002	0.5	4
7	0.4	0.9	0.4	9	0.1	0.0002	0.2	15
8	0.0	0.5	0	3M	0.02	0	0.2M	748
9	0.5	6	0.2	2K	0.2	0.9μ	177	70

in fan-out transactions, and so on. A large number of fan-in transactions impact the Mempool, but minimise the UTXO set.

2. **Fan-out.** The fan-out pattern involves one or two addresses sending funds to many addresses, as shown in **Clusters 0, 5, 8 and 9**; the total number of transactions in these clusters is about 165K. These transactions increase the UTXO set. This pattern was dominant in the clustering results; it resulted in multiple clusters for fan-out transactions that differ in features other than R. A low value for *CDD* indicates a fast movement of coins. Note that Cluster 0 includes transactions that have a single address sending small amounts to more than 3K addresses.

3. **Unable-to-decode.** With 425K transactions, **Cluster 7** includes the largest number of transactions. The distinct feature of most of these transactions is a one-to-one mapping: one address sending to a single output that cannot be decoded. Moreover, the fees paid for these transactions (which are collected by miners since the output cannot be decoded) equal the default fee value of 0.1 mBTC per KB. Another feature of this cluster is the zero value for *CDD* (and low P), which indicates rapid movement of bitcoins.

4. **Dust.** The final motif of the analysed spam campaign is the dust transactions we had previously discussed. **Cluster 7** contains non-spam transactions; normal transactions are matched to this cluster since they look similar to unable-to-decode transactions (low values for most features). It is not straightforward to visually

inspect the cluster samples and determine if they are indeed spam. Therefore, we parse the transactions in this cluster to determine which of them fit our definition of dust spam. We explain in a later section how we parse the results to find dust spam transactions.

5. **UTXO cleanup. Clusters 1 and 3** include "clean-up" transactions, created by miners to collate spam transactions to minimise the UTXO, thereby decreasing the spam impact on the network. The output addresses value of these transactions may be zero, meaning that all the inputs are collected as fees by the miner who includes the transaction in a block. Clean-up transactions include "Brain wallet" addresses (discussed earlier). These two clusters are not categorised as spam, and the transactions are a consequence of the spam campaign. The number of inputs to these transactions range between 1K and 5K (resulting in a large standard deviation).

Note that clusters 5 and 8 contain few transactions due to their unusually high P. Cluster 8, which contains only two transactions, is indeed interesting and earns its unique cluster: along with high P, the values of these transactions are around 2,500 and 3,995 bitcoins (that is almost $0.6M and $0.96M in respectively). Both transactions include a generous fee of 0.002 BTC.

In summary **Clusters 0, 2, 4, 6, 7, and 9** correspond to our definition of Bitcoin spam, including dust transactions and unusual ratios, while clusters 1 and 3 are a consequence of spam and not-spam motifs.

4.5.3 Validation

It is important to note that we lack an external source to create ground truth for our results. Without a labelled data set, or a third-party spam list, we cannot measure the clustering results to be spam more accurately than matching the results to our definitions of spam.

In order to find dust transactions, we check if P is low (less than 57M) and whether the transaction creates any outputs of 0.1 mBTC (about $0.02), which is the default fee value. We consider this a conservative estimate of the dust transactions involved in the spam campaign, and at the same time we consider the 0.01 BTC normally involved in dust checks to be too large.

We also applied clustering to transactions that occurred in the pre-spam epoch[6], between June 24 and July 7 (after filtering for dust and unusual ratios). The results are discussed in the next section, where we see a difference in the intensity of motifs before and during the spam epoch. This validates our clustering results: we find that the centroids obtained from training k-means, using the spam epoch data, can also detect spam patterns in non-spam epochs.

4.6 Impact on Bitcoin

We now describe the effects of spam campaigns on the Bitcoin network, especially on users who send non-spam transactions, and the miners. For the users, we measure the change

[6]We used the cluster centroids obtained from clustering transactions in the spam epoch to cluster transactions in the non-spam epoch.

in transaction fees and transaction delays (i.e., the time between when we first observe a transaction in the Mempool and when the transaction is committed to the blockchain). A large amount of spam is likely to increase the backlog of unconfirmed transactions. As a result, transactions are delayed for longer time periods. With more intense competition, senders pay higher fees, in the hope that their transactions will be included sooner. For the miners, we measure the corresponding increase in the block reward.

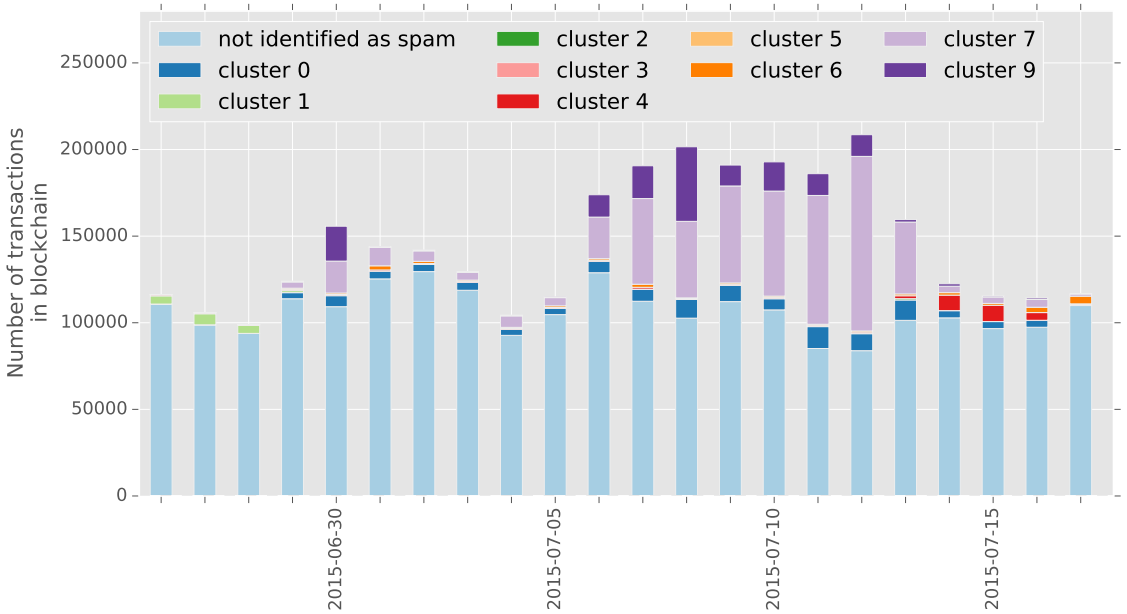


Figure 4.1: A stacked bar chart showing the number of transactions per day in the blockchain. Note that the spam period is from July 7 to 17.

Figure 4.1 shows the clustering results in the non-spam and spam epochs. Note that in the pre-spam epoch (before July 7), clustering results show Cluster 1 transactions (UTXO-cleanup motif). This does not mean that miners were cleaning up spam; these transactions are similar to UTXO-cleanup transactions in terms of high I and low O, and similar P values.

To highlight periods of the spam campaign, we measure the number of unconfirmed transactions in the Mempool, which indicates the amount of backlog in the network. Every minute, we take a snapshot of the Mempool and count the number of unconfirmed transactions. We take the average on a daily basis and plot the result in Figure 4.2.

Each major spike in the graph refers to a period of significant backlog. The first spike, which happened between July 7 and 17, corresponds to the spam campaign in our study. There are sporadic spikes between July and August, but we do not have sufficient insight on the cause. Finally, a spike appeared around September 13, when an anonymous group conducted another stress-test on the network with their "money drop" (as discussed earlier). As a result, a large number of transactions were created to compete for the free bitcoins, although only a few of them would be included in the blockchain eventually. Such a deluge of transactions caused the second backlog in Figure 4.2. We do not, however, consider these transactions to be spam.

Focusing on the mid-July spam period, we next examine the number of transactions that were committed to the blockchain. We are interested in how each block allocates its scarce 1 MB of real-estate space to spammers and non-spammers.

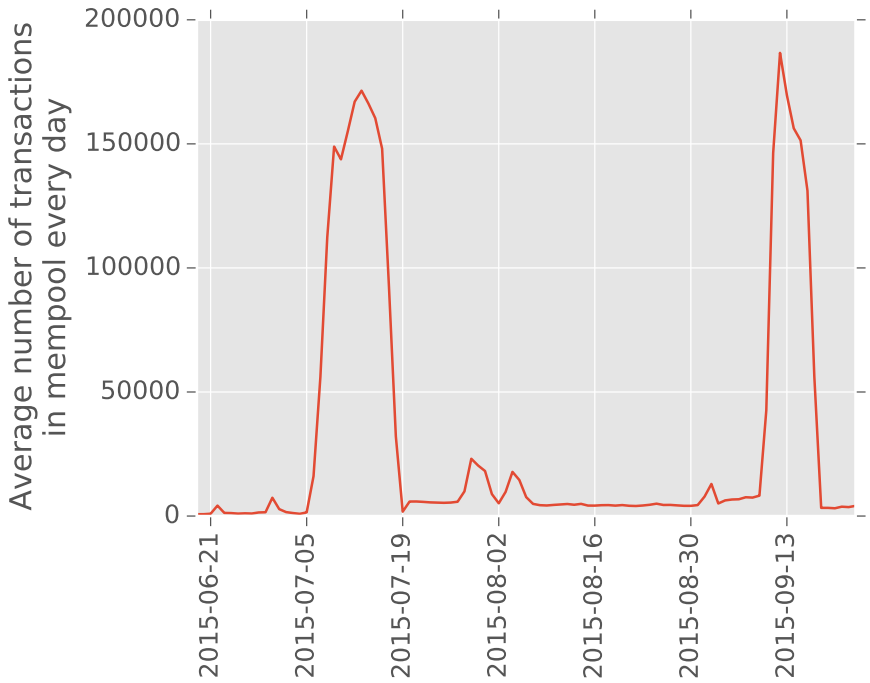


Figure 4.2: The average number of unconfirmed transactions in the Mempool every day.

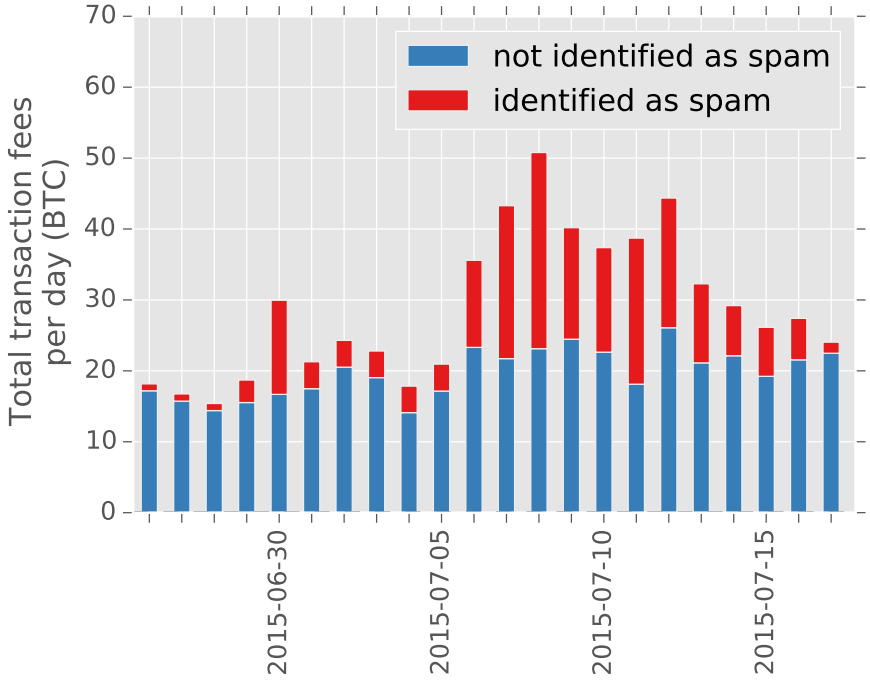


Figure 4.3: A stacked bar chart showing the total amount of transaction fees every day.

As shown in Figure 4.3, the number of transactions surged during the spam epoch. Between a quarter to half of the daily transactions have been identified as spam.

As a baseline comparison, we also show that the number of spam transactions before the spam period is significantly lower, with the exception of June 30. Based on anecdotal evidence, some users were attempting to stress-test the Bitcoin network on a small scale, which resulted in a brief rise in spam[7].

For the non-spammers, the spam period was a time when both transaction fees and delays were higher than normal. We show the comparison in Figures 4.4 and 4.5. On average, the delays in processing non-spam transactions increases by 7 times, from 0.33 to 2.67 hours. Likewise, the average non-spam transaction fees also surged, increasing from 45 to 68 Satoshis for every byte of transactions (or from $0.11 to $0.17 per kilobyte of transactions)–an uptick of 51%.

While non-spammers suffered, miners slightly benefit from the fee hike. As shown in Figure 4.3, miners were earning twice their normal fee-based revenue during the mid-July spam period, as compared with the non-spam period. However, even on days with maximum fees, the amount of extra mining income from fees was less than 1% of the block reward (which is 25 BTC per block, and about 3,600 BTC per day). The total transaction fees that spam transactions paid amounted to 201 BTC (or about $49,000) over a 10-day time period, a modest sum that caused a rather noticeable disruption to the network.

4.7 Discussion

The spam campaign happened when the Bitcoin network was divided on a critical component of the protocol: the block size limit of 1 MB. The result was a fork between two camps: some want to raise the limit while others refuse to alter the rules set by Satoshi Nakamoto (Bitcoin's creator). We can speculate that the spammer was motivated to launch a DoS attack to demonstrate the fragility of Bitcoin's resilience if the block size limit is not raised. This is supported by an earlier spam campaign, where an online Bitcoin wallet service claimed responsibility under the pretext of "stress testing".

With regards to the methodology proposed in this chapter, we do not suggest that this model can be used to prevent Bitcoin spam completely, nor should it be used as such. It was used to measure and analyse spam after the fact, without the spammers being aware that they are being measured. Spammers can learn from this research what heuristics and features we used, to alter their motifs and adapt accordingly.

Although we used dust checks to validate our results, this is not a foolproof measurement to accurately validate spam in Bitcoin. It is trivial to create a transaction that does not generate any dust outputs.

However, if a transaction does not create dust, then the clustering algorithm matches that transaction to a cluster that highlights other features, particularly differences in ratio. We use dust validation when there is a possibility a cluster contains normal transactions. Since we defined unusual fan-out ratios to constitute spam transactions (and they are gathered in distinct clusters), along with our conservative measurement of dust, we believe that the results we had shown earlier provide a good estimate of the July spam campaign.

[7] `http://motherboard.vice.com/read/wikileaks-is-now-a-target-in-the-massive-spam-attack-on-bitcoin`

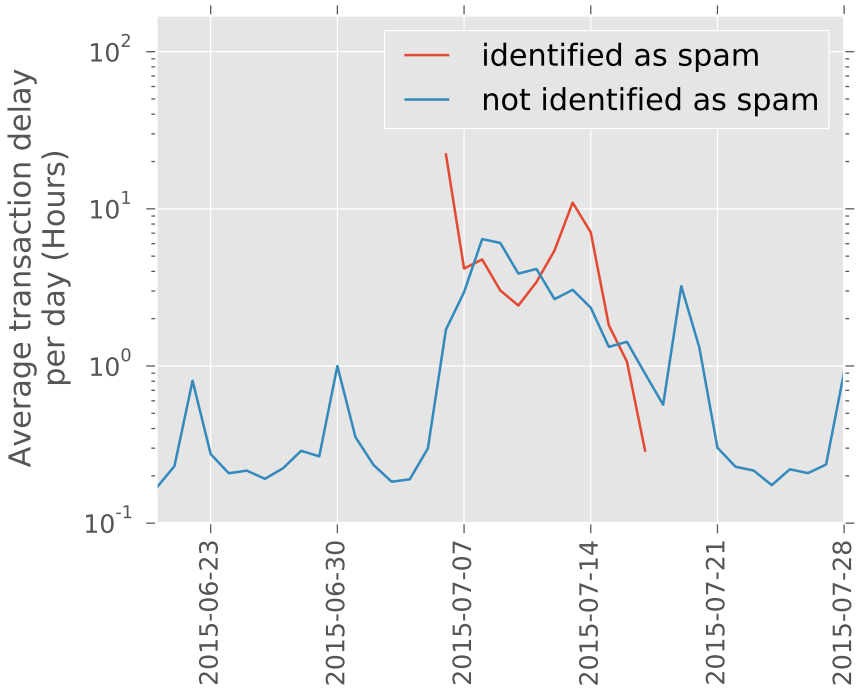


Figure 4.4: Average transaction delay between when a transaction appears in the Mempool and when it is committed to the blockchain.

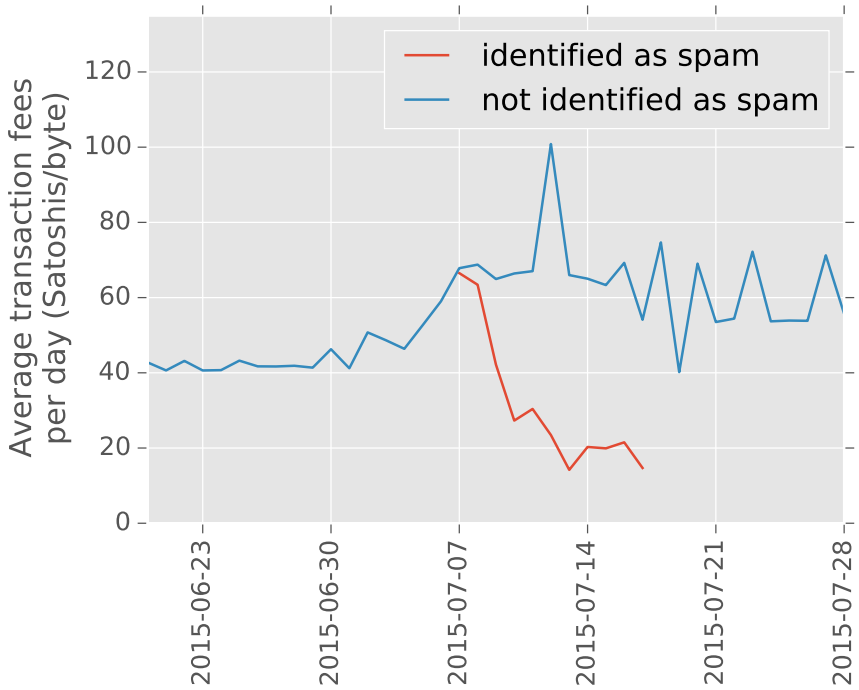


Figure 4.5: Average transaction fees per transaction per day. Note that the fees are normalised against the size of each transaction.

It is important to note that the July 2015 spam campaign on Bitcoin would be infeasible on many altcoins[8] since they may use a different model for transaction fees. For example, Litecoin charges the `mintxfee` for *each* small output. Bitcoin could adopt a similar model, or a dynamic model for fees, possibly using clustering results to observe spam patterns, and change `mintxfee` accordingly.

4.8 Related work

In their Bitcoin SoK paper [28], Bonneau *et al.* highlight open research questions and discuss issues with Bitcoin regarding stability and scalability, ranging from Bitcoin forks and network analysis to incentivising correct behaviour and adding resilience with proposed changes. The authors highlight *penny flooding*, as discussed in [91], which is related to our discussion of dust transactions. In the Appendix of the extended version of their SoK paper, the authors discuss in more detail Bitcoin's stability and transaction validity. The extended version includes a discussion outlining options to overcome the drawbacks of maintaining the entire UTXO to process new transactions: using a *statefile*, updated incrementally with new data, it is possible to more efficiently retrieve transactions for verification using a transaction's hash in $O(log\ M)$, where M is the number of unspent transactions. It also possible to further minimise the data structure required to validate transactions using hash-based authenticated data structures as proposed in [81].

Becker *et al.* [22] discuss the possibility of denying service to the Bitcoin network using a virtual protest: protestors join forces to collectively execute a DoS attack by overwhelming the network and depleting precious block space (using transactions that are much larger than normal Bitcoin transactions). If a protest is underway, this could frustrate non-protestors and decrease faith in the resilience of Bitcoin to process transactions in a timely manner (Bitcoin's main features include processing payments in minutes, as well as low transaction fees). This attack was called "Occupy Bitcoin" by Kroll *et al.* [74].

Our clustering approach is different from previous work aiming to find patterns in Bitcoin: we strip away identifying information (such as *txid*, addresses, etc.), and cluster transaction features in order to determine patterns, rather than linking transactions together to de-anonymise users. For example, Meiklejohn *et al.* [85] cluster transactions based on determining shared authority properties, while using transaction features similar to those used here.

Other empirical research on detrimental effects on the Bitcoin network measures the lifetime of Bitcoin exchanges [89], through analysing daily transaction volumes and how exchange breaches affect survival time. Running a profitable exchange logically results in a more lucrative target, hence a breach is more likely which leads to the eventual shutdown of an exchange. Another approach is to parse online forums to obtain data indicators of possible attacks on the network, as was done in [117].

4.9 Conclusion

This chapter presented an empirical study of a spam based "stress test" DoS attack against Bitcoin. Using our clustering based approach we find that 385,256 (23.41%) out

[8] Alternative cryptocurrencies (called altcoins) are similar to Bitcoin that utilise different protocols and have their own blockchains.

of 1,645,667 total Bitcoin transactions were spam during a 10-day period at the peak of the spam campaign. We also show that this attack had a negative impact on non-spam transactions, increasing average fees by 51% (from 45 to 68 Satoshis/byte) and processing delay by 7 times (from 0.33 to 2.67 hours). This shows that an adversary who is willing to spend modest amounts of bitcoin (we estimated $49,000) to pay higher fees, can DoS Bitcoin. Follow up DoS attacks against Bitcoin have used other methods, such as "money drops", and transaction malleability to degrade the operation of Bitcoin. We point out that changes to Bitcoin's minimum fees could mitigate some of the spam motifs we witnessed. Our results show that exploration into Bitcoin transaction spam filtering techniques, and other Bitcoin DoS mitigation approaches, merit further investigation.

Chapter 5

Offline payment protocols

In this chapter, we include the details for the offline payment protocol we created for a project, called *DigiTally*, in which we tried to use short message authentication protocols to produce a payment protocol that can be used offline to finalise payments. The motivation was to create a resilient payment protocol, that is not susceptible to issues raised in Chapter 2, and does not require network coverage to exchange messages with a centralised online server.

To do that, we designed the protocol to preserve the integrity of transactions, by creating short authentication codes that can be exchanged between participants. Moreover, similar to the social trust proposal (Chapter 3) and micropayment schemes (Section 2.4.2), we aim to provide means for decentralised verification of payments to increase performance and reduce points of failure.

The content of this chapter is adapted from our paper *SMAPs: Short Message Authentication Protocols*, by Khaled Baqer, Johann Bezuidenhoudt, Ross Anderson, and Markus Kuhn, which appears in the Proceedings of the International Workshop on Security Protocols 2016 [18].

5.1 Summary

There is a long history of authentication protocols designed for ease of human use, which rely on users copying a short string of digits. Historical examples include telex test keys and early nuclear firing codes; familiar modern examples include prepayment meter codes and the 3-digit card verification values used in online shopping. In this chapter, we show how security protocols that are designed for human readability and interaction can fail to provide adequate protection against simple attacks. To illustrate the problem, we discuss an offline payment protocol and explain various problems. We work through multiple iterations, or "evolutions", of the protocol in order to get better tradeoffs between security and usability. We discuss the limitation of verifying such protocols using BAN logic. Our aim is to develop human-friendly protocols that can be used in constrained offline environments. We conclude that protocol designers need to be good curators of security state, and also pay attention to the interaction between online and offline functions. We suggest that delay-tolerant networking might be a future direction of evolution for protocol research.

5.2 Introduction

Mobile payment systems have transformed the life of people in less developed countries (LDCs), bringing a convenient means of exchange and store of value to people living far from any conventional banking service. For rural people, phone payments enable everything from pensions to farm subsidies to be paid directly and efficiently, reducing the possibility for corruption and extortion. Even for people living in large cities, phone payments have greatly cut the cost of financial services. A further key application is remittances: migrant workers who have moved from the countryside to the city can send cash home to relatives. However, the currently deployed systems, such as M-Pesa in Kenya, use SMS or USSD as their carrier, and they stop when the network does. This leaves hundreds of millions of the world's very poorest people stranded. Living off-network in mountains, forests and small islands, they still have to use cash for transactions in their village, and are fully exposed to the depredations of dishonest officials.

There are four major limitations of current phone payments. First, both merchant and customer have to be online, so they don't work in areas without network coverage. Second, the cost of the SMS is unnecessary when half the customer's transactions are with the same village merchant. Third, existing systems are mostly locked in to a particular telco and bank as they rely on SMS or USSD. This lock-in increases costs and prevents inter-scheme payments, which is a problem for migrant workers. Fourth, most payment systems operate online. Resilience is not considered thoroughly, at least not beyond providing redundant servers to process requests. When networks go down because of power cuts or congestion, or rural networks close at night because base stations depend on solar panels and have no batteries, payment services cease.

Electronic purses for offline payments exist in the academic literature, and are also fielded, whether as standalone systems or EMV extensions (e.g., Net1 [3] and Germany's Geldkarte[1]). However, they have not been implemented by LDC mobile providers, who can install SIM-toolkit applets. There are both engineering and business issues; we are less concerned with the latter here, as they vary widely between countries.

The engineering issue is that existing purse systems are designed for complex messaging between the purses (e.g., between the purse in a smartcard and another purse embedded in a parking meter or ATM) while phone-to-phone payments conducted in the absence of a network or direct link (NFC, Bluetooth, infrared, etc.) must by default rely on users copying numbers between their phones. The protocols must therefore be redesigned for usability, which means minimising the number of digits that need to be typed while supporting robust error handling and recovery.

In this chapter, we show how security protocols that are designed for human readability and interaction can end up vulnerable to simple attacks. To illustrate the problem, we discuss our attempt to design an offline payment protocol, and the various problems we had to deal with. We present multiple iterations, or "evolutions", that the protocol went through as we sought to get reasonable tradeoffs between security and usability. Our target demographic is phone users in LDCs, many of whom are illiterate.

[1] `https://en.wikipedia.org/wiki/Geldkarte`

5.3 System model

The concept of operations is similar to existing phone payments, except that payments also work when the phone is offline. Sam is a GSM SIM card issuer who may also operate the offline payment system described in this chapter, which allows payments between SIM purses in areas with intermittent network coverage. In the following examples, Alice will play the role of a paying customer and Bob will be the shop owner receiving payment, but these roles can be reversed. Sam issues SIM_A to Alice and SIM_B to Bob.

The SIMs of Alice and Bob interact with each other only via human decimal number entry and comparison. Therefore, this protocol has to ensure that SIM_A and SIM_B agree on their transaction history, in spite of the low entropy of the manually-executed challenge–response round trips. The SIMs send records of past payments back to Sam whenever they detect network coverage, for reconciliation between their bank accounts.

In some applications, the network may be intermittent rather than absent, and so our protocols enable payment networks to support delay-tolerant authentication (see Section 5.5.2).

Where both Alice and Bob have smartphones that can communicate directly (Bluetooth, Wi-Fi, NFC, etc.), a very capable offline payment system can be built with conventional tools. However, many poor people use simple GSM phones that cannot communicate with each other directly in the absence of a GSM network. Then the only way for SIM-toolkit applets to communicate across phones is that their users copy sequences of displayed decimal digits. Numerical transactions are already familiar from other systems such as airtime purchases and prepayment utility meters. It is well known, for example, that even illiterate people can cope with twenty-digit sequences provided these are arranged as five groups of four digits, which is enough for a 64-bit ciphertext, and is used in prepayment meters [8].

5.4 Design evolution

The following setup is required in each iteration of the protocol. SIM_A and SIM_B are identified in the protocols by unique, human-recognisable decimal numbers A and B respectively, typically their phone numbers. Sam embeds an individual symmetric 128-bit private key K_A into SIM_A and K_B into SIM_B, which are tamper-resistant to some extent. (Sam could generate these keys from a master key K_S using a key-derivation function, such as $K_A = h_{K_S}(A)$, $K_B = h_{K_S}(B)$). These per-card keys are each known only to Sam and to one single SIM card.

5.4.1 Basic protocol

The basic payment protocol proceeds as follows[2]:

1. Alice agrees to pay Bob X and each of them enters both this amount and the other party's phone number into their phones.

[2]The steps performed by the users include the agreement on the amount X, the visual or verbal exchange of codes as displayed by the device, and the entry of codes into a device. Generating nonces and verifying MACs are performed by the SIMs. This applies to subsequent versions of the protocol.

2. Bob chooses a 4-digit nonce N_B and forms a 4-digit MAC C (using the shared secret key K) of B and X. He tells Alice the values

$$(N_B, C) \quad \text{where} \quad C = \mathsf{MAC}_K(B, A, X, N_B) \bmod 10^4 \tag{5.1}$$

with MAC being a 64-bit message-authentication-code function.

3. Alice verifies C. She now believes that she and Bob agree on the amount X and the payer and payee phone numbers A and B (with probability 0.999).

4. Alice authorises the transaction by entering her PIN; her purse decrements its balance by X and generates a 4-digit nonce plus a 4-digit MAC to bind her name and nonce to the data contained in the challenge C:

$$(N_A, R) \quad \text{where} \quad R = \mathsf{MAC}_K(A, N_A, C, N_B, B) \bmod 10^4 \tag{5.2}$$

5. Bob enters N_A and R into his purse, and checks it increments by X.

Verification. We then analyse this protocol via the Burrows–Abadi–Needham (BAN) logic [29]. We idealised the protocol as:

$$B \longrightarrow A : \{B, A, X, N_B\}_K \qquad (= C)$$
$$A \longrightarrow B : \{A, N_A, C, N_B, B\}_K \quad (= R)$$

We wished to prove that Bob the shopkeeper should trust the payment amount X, i.e., $B \mid\equiv X$. This can only be deduced using jurisdiction rule, for which we need $B \mid\equiv A \mid\Rightarrow X$ (B believes A has jurisdiction over X) and $B \mid\equiv A \mid\equiv X$ (B believes A believes X).

The former follows from our trust in the software, which has the property that it only creates ciphertexts that start with its own identifier. The value X is contained in the challenge C (but see section 5.4.2). The latter, that $B \mid\equiv A \mid\equiv X$, must be deduced using the nonce verification rule from $\sharp R$ (R is fresh) and $B \mid\equiv A \mid\sim X$ (B believes A uttered X).

Now $\sharp R$ follows from the fact that $R = h(K; A, N_A, C, N_B, B)$ and contains the nonce N_B Bob just generated. B believes A uttered X from the software constraint already mentioned.

In our original design, we did not include N_B in Bob's challenge, and as a result could not get the protocol to verify. This is in effect what happens with EMV, which in consequence is vulnerable to the preplay attack [27]. In our initial design, we also wondered whether C should contain Alice's phone number too: $C = \mathsf{MAC}_K(B, A, X, N_B)$. The BAN analysis showed this is superfluous. However, we include it for error-detection, as discussed later.

Despite the fact that the protocol verified, there was an attack.

5.4.2 Evolution 1: Eliminating narrow pipes

A crooked merchant Bob can perform the following attack against Alice:

1. Alice agrees on price X and Bob receives Alice's phone number A.

2. Bob now chooses a higher price X'.

3. Bob then repeatedly feeds X and X' to his SIM_B, which will generate a new nonce N_B each time and output a MAC C. Bob continues until he finds a colliding pair with the same MAC (X, N_B) and (X', N'_B) such that

$$\mathsf{MAC}_K(A, X, N_B, B) \equiv \mathsf{MAC}_K(A, X', N'_B, B) \equiv C \bmod 10^4.$$

   This will take just a few dozen trials for a 4-digit MAC.

4. Bob aborts all the trial transactions except for the last one for (X', N'_B).

5. Bob then gives (N_B, C) to Alice, but asks his own SIM_B to proceed with the last one, using N'_B and X'.

This way, Alice makes a payment for X, but Bob receives one for $X' > X$, violating conservation of money.

The vulnerability is that R only includes C, not X, and while C depends on X, C does not have enough entropy to prevent an online collision attack by Bob against SIM_B that allows X' to replace X. Once this is noticed, the fix is easy: include the amount X in the input to the payment MAC R, rather than the challenge C.

Such attacks are beyond the scope of BAN, which was only intended for protocols involving cryptographically long nonces and MACs, and does not keep track of guessing entropy or collision risks. Some more modern cryptographic verification tools (e.g., CryptoVerif [25]) can quantify such probabilities.

Anyway, the repaired protocol now runs:

1. Alice agrees to pay Bob X.

2. Bob chooses a 4-digit nonce N_B, forms a 4-digit MAC C with B and X, and sends Alice:

$$(N_B, C) \quad \text{where} \quad C = \mathsf{MAC}_K(B, A, X, N_B) \bmod 10^4 \tag{5.3}$$

3. Alice verifies C to ensure they agree on payer, payee and amount.

4. Alice authorises the transaction by entering her PIN; her purse decrements its balance by X and generates a 4-digit nonce plus a 4-digit MAC:

$$(N_A, R) \quad \text{where} \quad R = \mathsf{MAC}_K(A, N_A, X, N_B, B) \bmod 10^4 \tag{5.4}$$

5. Bob enters N_A and R into his purse, and checks it increments by X.

The verification proceeds as before, although by now we might place less reliance on a BAN-logic verification of a short message authentication protocol.

5.4.3 Evolution 2: Transaction chaining

The revised protocol still raised security concerns:

1. Bob could try to add money to his SIM card by faking transactions with fake customers and just guessing the response R. Each attempt will succeed only with probability 10^{-4}, and repeated attempts can be blocked by a retry counter. We can also mix up R with N_A. However, the detailed design is far from trivial. For

example, Bob could connect SIM_B to a PC to automate an attack, and interleave guessing attempts with real payments from a customer SIM he controls. In the worst-case attack, Bob is the local payment service agent, and has hundreds of customer SIM cards in stock. We need to make the fraud probability acceptably low without making the authorisation code too long, or otherwise making operations overly complex or fragile.

2. Alice can similarly fake a transaction with probability 10^{-4}, but this may be less of a concern in practice, as Alice cannot repeat thousands of transaction attempts against SIM_B without collaboration by Bob; while a colluding Bob could just run the attack without Alice (as above).

3. Bob can also try to fake transactions with real customers A, by keeping a record of their $\mathsf{MAC}_K(A, N_A, X, N_B, B)$ replies. In such a fake transaction, Bob can choose A and N_A, and if the real Alice has already paid n times in the past for a regularly bought item of fixed price X, then Bob has enough data to be able to look up a valid pair (N_B, R) to complete a fake transaction with probability $n \cdot 10^{-4}$.

The last attack is of particular concern if Alice makes daily purchases of the same price X for years, as the probability of Bob being able to fake a transaction response R from SIM_A approaches one. Our solution is to establish a payment session between Alice and Bob that maintains additional shared entropy stored in both SIM_A and SIM_B. We include this state in the MAC inputs, such that knowledge of past MAC responses no longer helps Bob guess future ones. We also replace N_A and N_B with MACs of the transaction data that only the bank Sam can verify. Then even if a fake transaction is accepted by a SIM, it will still be spotted when one of the parties eventually uploads it to Sam.

Most phone payment transactions in LDCs are to familiar recipients. A villager will do most of their shopping at the one village store; a migrant worker will make most remittance payments to his wife or perhaps his mum back in his home village. So the obvious next evolution is to look for ways in which subsequent payments can be made easier. This has been a familiar strategy in established online banking systems for about 30 years now.

If the security of payments authenticated using short codes must depend on maintaining as much shared security state as possible, then why not use a hash chain of all transactions in the current session, rather than just the current transaction context?

Let a payment session be a hash chain maintained in both cards. Its state is kept in each card as a 256-bit value S_i, along with a transaction counter i. When A and B start a new series of transactions, both their SIMs initialise this session as a hash state of

$$T_0 := (A, B) \tag{5.5}$$
$$S_0 := H(0, T_0) \tag{5.6}$$
$$i := 0 \tag{5.7}$$

where H is a collision-resistant hash function with 256-bit output (e.g., SHA-256 or SHA-3). The transaction counter i records how many payments have been committed in this session. Alice and Bob now should have the same value of S_0.

SIM_A and SIM_B also set up a shared transaction key K_{AB}. At this stage we can assume it is simply derived from their phone numbers using a key derivation function and

the universal shared secret K: $K_{AB} = h_K(A, B)$. In Section 5.5 we discuss options for mitigating the risk of a shared master secret.

Once the payment session is established, Alice can pay Bob as follows:

1. Alice and Bob agree on a price X.

2. Alice and Bob then select the payment session to be used, and their SIMs retrieve not just A and B but also the hash chain values (i, S_i).

3. SIM_A checks that X is within limits agreed with Sam, and that Alice's on-card purse value M_A has enough funds for the transaction, presumably $M_A - X \geq 0$. It aborts the transaction otherwise. Likewise, SIM_B checks whether both X and its purse value $M_B + X$ are within limits agreed with Sam, and aborts if not. (Transaction limits agreed with Sam may be variable, and depend on for example X, i, A, or B.)

4. Bob's SIM_B generates and displays a challenge of $n_{\text{c},1}$ deterministic digits that Alice and Sam can verify, $n_{\text{c},2}$ deterministic digits that Sam can verify, as well as $n_{\text{c},3}$ digits of random entropy to help ensure that payment sessions do not repeat – an $n_{\text{c}} = (n_{\text{c},1} + n_{\text{c},2} + n_{\text{c},3})$-digit decimal challenge message

$$\begin{aligned} C_{i+1} = E^{n_{\text{c}}}_{h_{K_{AB}}(S_i)}[&(\mathsf{MAC}_{K_{AB}}(i, S_i, X) \bmod 10^{n_{\text{c},1}}) \parallel \\ &(\mathsf{MAC}_{K_B}(i, S_i, X, F_B) \bmod 10^{n_{\text{c},2}}) \parallel N_B] \end{aligned} \tag{5.8}$$

   which Alice copies into her phone. Here $0 \leq N_B < 10^{n_{\text{c},3}}$ is a number picked by SIM_B and $\parallel$ is concatenation of decimal digits. $E^{n_{\text{c}}}$ is a pseudo-random permutation over $\mathbb{Z}_{10^{n_{\text{c}}}}$ (see [24]), to ensure that adversaries cannot be certain about the function of individual digits without knowing K_{AB}. F_B is optional other information that can be included and that SIM_B will eventually transmit to Sam (but not to SIM_A) such as M_B or a timestamp, added as an entropy source and as forensic evidence of disputed transactions.

5. Alice's SIM_A then calculates an $n_{\text{r}} = (n_{\text{r},1} + n_{\text{r},2} + n_{\text{r},3})$-digit response

$$\begin{aligned} R_{i+1} = E^{n_{\text{r}}}_{h_{K_{AB}}(S_i, C_{i+1})}[&(\mathsf{MAC}_{K_{AB}}(i, S_i, X, C_{i+1}) \bmod 10^{n_{\text{r},1}}) \parallel \\ &(\mathsf{MAC}_{K_A}(i, S_i, X, C_{i+1}, F_A) \bmod 10^{n_{\text{r},2}}) \parallel N_A] \end{aligned} \tag{5.9}$$

   containing $n_{\text{r},1}$ digits that Bob can verify, $n_{\text{r},2}$ digits that Sam can verify, and $n_{\text{r},3}$ digits N_A chosen by SIM_A. F_A is optional other information that SIM_A will record and eventually transmit to Sam, such as M_A or a timestamp.

6. SIM_A then updates its non-volatile state and on-card transaction records:

$$M_A := M_A - X \tag{5.10}$$
$$i := i + 1 \tag{5.11}$$
$$T_i := (S_{i-1}, X, C_i, R_i) \tag{5.12}$$
$$S_i := H(i, T_i) \tag{5.13}$$

7. SIM_A finally displays its response message (5.9).

8. Bob types R_{i+1} into his phone, which reports the success or otherwise of the transaction.

The point of hash chaining is twofold:

- Even if someone gets lucky and guesses a correct authentication code R_i for an individual transaction, this will still leave the underlying security state S_i inconsistent between Alice and Bob, between whom subsequent transactions will then fail with high probability.
- The entropy of the security state S_i makes it less likely that a query to $\mathsf{MAC}_{K_{AB}}$ is ever repeated. This reduces the risk that knowledge of past values of R and C can help an attacker to predict future such values, and use these to set up fake transactions.

There are various attacks to consider, for example Bob can complete a fake transaction on SIM_B with probability $10^{-n_{r,1}}$, or duplicate sessions to double-spend Alice's responses. To fine-tune such risks, we can vary the number of digits allocated in the exchanged messages C and R between the first transaction and later transactions in a session, for example:

	$n_{\mathrm{c},1}$	$n_{\mathrm{c},2}$	$n_{\mathrm{c},3}$	$n_{\mathrm{r},1}$	$n_{\mathrm{r},2}$	$n_{\mathrm{r},3}$
first transaction	1	1	3	6	3	3
later transactions	1	1	1	4	3	0

In addition, we need to limit the number of failed transactions:

- A SIM can keep a record of failed transactions and can ensure that only a fraction of all transaction attempts per session are allowed to fail. For example, a payment session can be terminated once 5 of the last 10 transaction attempts have failed. Such retry limits should be implemented only on a per-session basis, to reduce the risk of denial-of-service attacks, where an adversary deliberately exhausts a SIM-wide retry limit.
- If retry limits apply per session, we then also need to limit the total number of sessions that a SIM can participate in, to well below $10^{n_{r,1}}$, e.g., a few thousand. This way, Bob cannot iterate fake transactions across many non-existing customers.

The exact limits and number of authentication-code digits can be tuned based on risk analysis and usability studies. They could also be made variable, though the usability consequences would have to be tested carefully. Likewise, the content of F_A and F_B remain for further study.

5.5 Mitigating the risk of a shared master secret

5.5.1 Evolution 3: Group key scheme

If smartcards were perfectly tamper-resistant, then the initial design would be largely complete at this point. Smartcards are certainly more resistant than twenty years ago, when large-scale compromises of pay-TV cards were pretty well an annual occurrence. This is partly due to technological improvements such as side-channel countermeasures,

top-layer sensor meshes, and randomised place-and-route; and partly due to minimising the value that can be extracted by cloning a single card. In EMV, for example, the keys authenticate transactions on a single account, so the extractable value is typically in the low tens of thousands of dollars, rather than the millions that could be made from cloning a pay-TV card. As long as reverse engineering a card costs tens of thousands of dollars and the attacker needs perhaps a dozen identical cards to work with, this is probably enough.

The overlay SIMs used in our field trial are not certified to EMV standards, though the vendor assures us that an EMV compliant overlay product will be available in due course. So in the short term we might deploy a two-tier system in which merchants get a high-security EMV-certified SIM card containing a master key while customers get a medium-security overlay SIM card containing a derived key, namely their phone number or card serial number encrypted under the master key to provide a diversified key. This is the approach used for some 20 years in Net1 and for a decade in Geldkarte. It would thus have the advantage of being familiar to banks and their insurers, with a zero loss history and the comfort that comes from reusing existing standards and business processes. If anyone seeking to monetise a break of a card needs to extract money from merchants (as the other users are too dispersed and too poor) then the Geldkarte approach could make perfect business sense.

However, a substantial volume of LDC phone payments are migrant remittances, where neither the sender nor the receiver is a merchant.

A second approach is combinatorial keying, which was proposed in the 1990s in the context of satellite TV, where it was known as "broadcast encryption", and attempted to give each pay-TV subscriber a set of keys of which some suitable subset could enable her to decrypt each programme. If a card were cloned, then that particular subset could be blacklisted without affecting the great majority of other subscribers; only a small number of subscribers with the same subset of keys would have to be issued with replacement cards.

A similar idea can be adapted here, where the communications are many-to-many. For example, we can divide all users into $d = 100$ key groups, give everyone 100 out of 5050 keys, and thus require an attacker who wanted to impersonate any user to clone 100 cards rather than just one.

In more detail: Sam generates a set of shared 128-bit group keys, in the form of an upper triangular matrix of $d(d+1)/2$ keys $(K_{i,j})_{0 \le i \le j < d}$. (Sam could again generate all these from a master key K_s using a key-derivation function, such as $K_{i,j} = h_{K_s}(i,j)$).

Each interoperable SIM issued by Sam also contains a common private 128-bit key K_{g}. Sam uses K_{g} to assign each SIM to one of d key groups ($d \approx 10^2$), using a pseudo-random function applied to its phone number:

$$g_A := h_{K_{\mathrm{g}}}(A) \bmod d \tag{5.14}$$

$$g_B := h_{K_{\mathrm{g}}}(B) \bmod d \tag{5.15}$$

Sam stores in each SIM in key group g_A the d group keys

$$G_A = \{K_{0,g_A}, K_{1,g_A}, \ldots, K_{g_A-1,g_A}, K_{g_A,g_A}, K_{g_A,g_A+1}, \ldots K_{g_A,d-1}\}. \tag{5.16}$$

(and equivalently for SIMs in group g_B, etc).

SIM_A and SIM_B can now secure their message exchanges using the key

$$K_{AB} = \begin{cases} h_{K_{g_A,g_B}}(A,B) & \text{if } g_A \le g_B \\ h_{K_{g_B,g_A}}(A,B) & \text{if } g_A > g_B. \end{cases} \tag{5.17}$$

K_{g_A,g_B} or K_{g_B,g_A} will be available in both SIM_A and SIM_B, and both can use K_g to select it. Any other SIM_U picked at random (for reverse engineering) by someone who does not know K_g will contain it only with probability $2d^{-1} - d^{-2}$ (i.e., $2d^{-1}$ if $g_A \neq g_B$ and d^{-1} if $g_A = g_B$).

This is straightforward to do if each user gets a SIM card issued by a payment service provider that is also an MNO. Where the two are different, and especially if the payment service runs on an overlay SIM independent of the phone company's SIM, it is not so straightforward, because of the need to establish a trustworthy link between the user's phone number and her key group. This can be done by an online protocol if the phone is online as the overlay SIM is first fitted, but otherwise requires copying quite a few digits.

5.5.2 Evolution 4: Delay-tolerant Needham–Schroeder

In many applications of interest, both Alice and Bob will get network connectivity from time to time as they travel to town or through areas with mobile service. There is significant research in more general store-and-forward mechanisms or delay-tolerant networks for extending service in LDCs. Even in the few cases where one of the parties does not ever travel to town, connection can be established via a *data mule*. For example, if Bob is old and housebound, while his daughter Alice works in the big city and sends him money, a neighbour who travels past Bob's hut can take data to and from a point of network presence. This should let us establish authenticated key exchange, although it appears to be unexplored territory from the protocols community's point of view.

An initial idea is to turn the bug in the Needham-Schroeder protocol [96] into a feature. The fact that Bob has no guarantee of the freshness in Alice's initial exchange with Sam, and that she can therefore pass on a key packet to him a month or even a year afterwards, can be used to create a delay-tolerant version of the protocol. In order to preserve the original notation of [96], we reverse the roles in this iteration whereby Alice (A) is the merchant travelling to the city, and Bob (B) is the housebound villager who requires relayed messages from Sam (S). We assume that Alice and Bob have just done the first transaction in a new payment session, and want Sam's help to set up a K_{AB} that is present only in their SIM cards and thus cannot be compromised if any other card is reverse-engineered.

1. Alice catches the boat into town, and SIM_A detects a connection. SIM_A automatically uploads transaction logs to Sam (to update shadow accounts for reconciliation). She also sends a request to authenticate with any new counterparty, in this case Bob[3]:

   $A \rightarrow S : A, B, N_A$

2. Sam sends K_{AB}, via SMS or USSD, encrypted with the shared key K_{AS}:

   $S \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$

3. When Alice returns to the village, SIM_A can display, during transactions discussed below, the authenticated shared key:

   $A \rightarrow B : \{K_{AB}, A\}_{K_{BS}}$

[3]The notation here follows the original Needham–Schroeder protocol [96].

4. The final two steps of the Needham–Schroeder protocol are performed offline between the transacting parties, by showing authenticated nonces to obtain agreement:

$B \to A : \{N_B\}_{K_{AB}}$
$A \to B : \{N_B - 1\}_{K_{AB}}$

Note that an implementation would be likely to optimise this somewhat, for usability reasons. The final two steps can be reduced to challenges and responses of a few digits each using the methods in earlier sections, but the key packet $\{K_{AB}, A\}_{K_{BS}}$ is harder. If K_{AB} were a 128-bit key that alone would require copying 40 decimal digits. Users of prepayment utility meters (including illiterate users) regularly copy 20-digits strings that represent DES-encrypted commands, so perhaps 40 digits can be done as a very occasional procedure. More likely, an operator might take the view that 30 digits are enough; an 80-bit key would not be the weakest link. Again, this requires real testing in the field. Where users interact with merchants directly, the merchant can perhaps help (subject to worries about the merchant ripping off vulnerable customers). But where a helpful neighbour is acting as a data mule for an elderly and housebound Bob, and carrying messages scribbled on bits of paper, 40 digits may well be too much.

More research is needed about how to integrate online payments, offline payments, data mules, and delay-tolerant networking in general. The value of this research is not just to extend and enhance payment systems per se, but also all sorts of other electronically enabled services, from pay-as-you-go solar panels to agricultural subsidies and payments from aid donors.

To evaluate such systems we need to gain insight into what transactions are taking place online versus offline; which transactions are critical (e.g., large transactions or cash-outs); incentives for contacting back-end systems; incentives for carrying data on behalf of other users; and realistic threat models. Should we assume that everyone in a village would happily conspire to cheat a bank, or that the headman or store owner in a village might be happy to be responsible for the village's good behaviour, or that villagers might vouch for each other?

5.6 Conclusion

Phone payments have been transformative in Africa, South Asia and elsewhere, helping millions of the very poor to raise themselves from poverty, and helping the less poor too. Extending them to areas without network service will continue this process but will also give disproportionate benefit to the very poor as the existing networks have been rolled out first to the less poor regions. The Gates Foundation advertised for a technology to do this and we received funding from them to build a prototype.

The mission is to design an offline electronic purse system optimised for use in less developed countries. Unlike existing systems such as Geldkarte and Proton[4], our design aims to maximise usability in off-network transactions by minimising the number of digits a user has to speak, hear and type, while providing robust recovery mechanisms for the inevitable errors and making sure there is not any scalable attack that is large enough to care about. We discuss the prototype and initial usability evaluation in the next chapter.

[4]See `https://en.wikipedia.org/wiki/Proton_(bank_card)` and `https://web.archive.org/web/20110511165455/http://www.atosworldline.be/index/en_US/5118014/5126207/Proton.htm`

In this chapter we have set out the evolution of our proposed core payment protocol. There are a number of lessons for protocol researchers.

First, when we start dealing with authentication protocols with short strings of digits, our intuitions about security can fail us, and formal verification does not always help. Our first attempt at a protocol was vulnerable to a simple collision attack, because we used the entropy that was visible rather than all the entropy available. Worse, the defective protocol verified just fine using the BAN logic. That does not imply that BAN is useless (as we did find another design flaw) but just that it is not enough in this context. We leave to future work whether other verification tools can find such flaws.

Second, when dealing with very short authentication codes, transaction chaining can be useful, and is a reasonable next step once we have learned to use all the available entropy. It had already appeared a generation ago in EFTPOS systems in Australia, where MAC residue was used for key updating, but then apparently had been forgotten.

Third, when a system is offline part of the time, then the interaction between the offline component and the online one bears careful study. If much of the world will have to live with delay-tolerant networking, then it is time to start thinking about how to incorporate delay tolerance into the infrastructure, rather than its being an occasional hack that developers will often get wrong. This may seem counter-intuitive to some providers, given that banks in the developed world spent the first half of the 1990s ripping out offline capabilities from ATM systems, and the second half reducing merchant floor limits for credit-card transactions. Yet it is noteworthy that when they introduced EMV in the 2000s, offline capabilities reappeared, and they did so by placing carefully calibrated amounts of trust in largely tamper-resistant smartcards and terminals.

A lot of services are going to have to coexist with network outages. That means, we suggest, that the next challenge for the protocols community will be to work out ways to build support for delay-tolerant networking into the infrastructure. We hope the examples here will help bootstrap the discussion.

Chapter 6

Usability evaluation of offline payments

We built a prototype system for our project, *DigiTally*, that implements the offline payment protocol we discussed in the previous chapter[1]. This project aimed to create an offline payment system that works on cheap phones to alleviate the problems stemming from the lack of network coverage. Design goals included prioritising usability while avoiding unfamiliar hardware. Another design goal we aimed for is to create a simple and resource-efficient protocol that works on resource-constrained hardware, such as SIMs and feature phones (not smartphones). We also prioritised speed (quick results of transaction processing), and aimed to minimise or mitigate user errors.

We use overlay SIMs to load our DigiTally code into a tamper-resistant environment to safely execute cryptographic operations, and protect the DigiTally protocol. Using overlay SIMs allows us to have a portable solution that works on any phone (including smartphones), that can be inserted into a user phone so that the DigiTally application is accessible from a SIM menu. The process is familiar to mobile payment systems users since those systems are also accessible through SIM menus.

The content of this chapter is adapted from our paper *DigiTally: Piloting offline payments for phones*, by Khaled Baqer, Ross Anderson, Jeunese Adrienne Payne, Lorna Mutegi, and Joseph Sevilla, which appears in the Proceedings of the Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017) [20].

6.1 Summary

Mobile payments support a range of services in many less developed countries including everyday payments, migrant remittances, credit, tax collection, and welfare benefits. These services depend entirely on the mobile phone network as their carrier, so they stop where the network does. This leaves millions of the very poorest people stranded – people living in remote areas where there is little to no network service. It also leaves urban users at the mercy of network congestion.

We developed a prototype system, DigiTally, which lets users make offline payments by copying short strings of digits from one mobile handset to another. Offline payments

[1]The DigiTally project webpage is available at `https://www.cl.cam.ac.uk/~kabhb2/DigiTally/` and archived at `https://web.archive.org/web/20171008121605/https://www.cl.cam.ac.uk/~kabhb2/DigiTally/`.

are already used for electricity (both in prepayment meters and pay-as-you-go solar); can we extend them into a general-purpose payment system, to increase service resilience in the face of network congestion or outage, and provide service to currently excluded areas?

We report the results of a preliminary study with an early prototype of DigiTally, tested on participants from a university in Nairobi, Kenya. The code-sharing process presented a possible usability challenge. To explore this and other aspects of our first prototype, DigiTally was introduced to Kenyan participants in order to resolve any major issues before a later field trial.

We discuss the lessons learned from our field visits and initial usability evaluation; we hope that this contribution is helpful for researchers and policy makers interested in mobile payments and financial inclusion. We also present our findings and observations. We found that, although offline payments involve copying codes in both directions between the payer's phone and the payee's, the extra workload was acceptable to most users.

6.2 Introduction

Because of the strong positive effect on development, the Bill & Melinda Gates Foundation called for innovations that could increase the uptake of mobile payments[2]. One of the largest impediments is that current systems operate entirely online; both the payer and the payee have to be able to communicate with the payment system server for a payment to be completed. This excludes millions of people living in remote areas with no network service; such people make up 10-40% of the population depending on the country. It also makes payments harder in the event of network congestion (we have observed 30-second delays in down-town Nairobi). Additionally, where the payment service operator is not the same firm as the Mobile Network Operator (MNO), charges become an issue.

The main contribution is to describe a preliminary study that took place at Strathmore University in Nairobi, in September 2016. We set out to establish whether DigiTally was usable in three different environments: a coffee shop, a campus bookshop, and a cafeteria, and by students from a range of backgrounds. These students were experienced users of M-Pesa and thus able to compare it with DigiTally; we also got assessments from the checkout staff. Further contributions are as follows: we analyse a security technology and offer lessons learned from a preliminary usability study. This chapter should be of interest to researchers interested in development, and also to those interested in evaluating payment systems; both communities can benefit from our insights. Documenting the study in one publication should help them and others interested in this type of research.

We discuss the background and related work in Section 6.3. We describe the technology in Section 6.4, and we discuss our method and results in sections 6.5 and 6.6. We describe our observations in Section 6.7, and provide more discussion in Section 6.8. Finally, we present our conclusions in Section 6.10.

6.3 Background and related work

We discussed mobile payment systems in Section 2.3. That section is relevant to understand how mobile payments work and what principals are involved in processing transactions.

[2]Enable Universal Acceptance of Mobile Money Payments: `https://gcgh.grandchallenges.org/challenge/enable-universal-acceptance-mobile-money-payments-round-14`.

Jack and Suri discussed the economics of M-Pesa in [66], highlighting issues that affected system uptake, including liquidity and network reliability. Zimmerman and Baur discuss the challenges facing financial inclusion efforts [124], including network coverage and reliability, liquidity, complexity of user interfaces and payment processes, the lack of dispute resolution, and the lack of customer protection against fraud. Dupas *et al.* report that a significant proportion of the participants in their study listed fraud, unreliable services, and transaction fees as issues [49].

The goal of our work is to tackle the network coverage and unreliable services problems by processing payments *offline* reliably and deterministically, and to simplify the user interface as much as possible by mimicking familiar mobile payment systems. Furthermore, we aim to decrease transaction fees to encourage users to process transactions electronically. The purpose of this research was to investigate possible usability challenges to be tackled in subsequent prototypes. Offline payment systems have already been implemented, such as Geldkarte[3] in Germany and Net1/UEPS [3] in South Africa, but require dedicated devices or unfamiliar hardware that can be costly to replace if lost or stolen. Similar systems might perhaps be implemented on modern smartphones if both the payer and the payee had them. However, most users in less developed countries still use feature phones that do not have NFC, Bluetooth, or cameras; these phones cannot communicate data automatically in the absence of a network. We also minimise the assumptions we make regarding what features are available on users' phones.

Our goal is to design a system that operates without relying on such features, in case some of them are inoperable. For example, if we rely on a camera and it is damaged, the entire system is useless until the user has their camera fixed. By operating within these constraints, we are able to design a system that works on all low-end mobile devices; our project aims to provide a solution for the poorest demographics.

The use of feature phones emerged as a requirement during trips to rural Kenya. Users in Busia county, for example, specifically requested solutions that work on feature phones. One person commented: "*Don't give us [systems that work on] smartphones. We don't have those things and we don't know how to use them.*" When asked what kind of phones users owned, all but one person (in a group of more than 20), held up a feature phone; the non-feature phone was an old BlackBerry. Reliance on feature phones (called '*kabambes*' in Kenya) is due to both battery life and cost (in terms of price as well as maintenance). With such constraints, the only way to transfer information offline is for one phone to display it, and for a human to type it into the other phone (a similar approach used in device-pairing methods). Kainda *et al.* [71] looked at the tradeoff of usability and security with regards to different device-pairing methods, using out-of-band channels, as it applies to various device usage or capability restrictions. Their results show that typing strings (*"copy & enter"*) ranked first in terms of the aforementioned tradeoff. They also recommend that system designers take into account factors that affect that tradeoff including user conditioning, user motivation, security failures, and attentiveness. Moreover, transferring value by copying digits is well established in other applications, such as prepayment electricity meters [8].

[3]Geldkarte website: `https://geldkarte.de/`.

Figure 6.1: An overlay SIM (top) and a regular SIM (bottom). The top part of the overlay SIM can be peeled off and stuck on top of a regular SIM

6.4 Technology

An early design goal we set was that DigiTally must not require users to operate any unfamiliar hardware, and that the transaction flow should be as close as possible to the familiar one. This leads immediately to the challenge of programming feature phones: different operating systems make it difficult to implement and test applications for various models, leading to large costs for maintenance and support.

To minimise these costs within our constraints, the practical approach is to program the SIM card in the user's phone. SIMs were designed to host multiple applets in secure containers, which can prevent one applet from accessing other applets' data. A SIM provides a secure environment that we can control and that is compatible with all mobile devices adhering to the Global System for Mobile Communications (GSM) and smartcard interoperability standards. This means that we can program our system on a SIM, and insert it into any device that accepts a SIM. The user can move the SIM from one device to another, allowing for portability. SIMs also provide valuable built-in features, including atomic operations and rollback mechanisms, as well as the ability to store secure tokens and cryptographic keys in a tamper-resistant chip.

Feature phones normally have a single SIM slot already taken by the SIM issued by the Mobile Network Operator (MNO). MNOs do not generally let anyone else program their SIMs, but there is a new technology that can be used to circumvent this restriction. This is the overlay SIM (or sticker SIM): a SIM card only 120 microns thick that can be inserted between the regular SIM card and the phone. Figure 6.1 shows an overlay SIM and a regular SIM. Overlay SIMs were developed to support low-cost mobile roaming. They can also be used as a proof-of-concept prototyping environment, and to bypass MNOs' restrictions on devices if necessary. We used the programmable overlay SIM as a regular SIM, inserted in the single SIM slot. The same overlay SIM can also be stuck on top of an existing non-overlay SIM, and our system will work seamlessly on the device while allowing it to still work as a phone[4].

[4]Overlays SIMs are used in China for roaming purposes, and were used in Kenya by a local bank to provide financial services to their users and to contest the local phone-payment monopoly. Overlay SIMs work on top of standard carriers' SIMs.

6.4.1 Overview of DigiTally

We designed and developed DigiTally as a Java Card applet accessed through a user's phone (Figure 6.2a). This applet can be loaded on an overlay SIM or a regular SIM. We chose to use a regular SIM for this study (for reasons we discuss in Section 6.5.4 item 2). As discussed earlier, having an overlay-SIM ready applet enables deployment on feature phones even if the MNO chooses not to install DigiTally on their own regular SIM.

Rather than asking participants to use their own phones, we provided participants with dedicated phones (Nokia 130), preloaded with a SIM that included the DigiTally applet. The applet includes a menu that mimics existing mobile payment systems to capitalise on users' familiarity with them. The applet offers the user options such as `Balance`, `Send Money`, and `Receive Money` (Figure 6.2b). Figures 6.3 and 6.4 illustrate the steps required to complete a transaction.

The icon for the applet displayed in Figure 6.2a is enforced through the mobile OS. When a single SIM is present, selecting the SIM icon will display the applets that reside on that SIM. When multiple SIMs are present, such as in the case of using an overlay SIM on top of a regular SIM, selecting the SIM icon directs the user to two different menu options that represent the applet names in the two SIMs. Note that there could be more than one menu option displayed after selecting the SIM icon, if one or both of the SIMs offer more than one applet.

There are two main differences between traditional mobile payment systems and DigiTally. The first difference is that the SIM in an offline payment system stores the *balance* as a value counter, and the payment protocol changes this local balance in the SIM, whereas a traditional mobile payment system merely sends requests to a backend system to process operations on the user's balance. The second difference is the `Receive Money` option: in the absence of an online payment server, the recipient must be involved in completing a transaction. In traditional mobile payment systems, the recipient is not actively involved; they merely get an SMS saying how much money has arrived and from where. In offline payments, both phones must be involved in a transaction. The protocol is designed so that the parties learn immediately whether the transaction has failed or completed. Note that DigiTally does not require data to be sent over the network and can, therefore, work completely *offline*.

For the purposes of this chapter, we focus on the codes exchanged by the users to complete transactions. We include below a summary and overview of the protocol discussed in Chapter 5 and as implemented in the DigiTally applet.

6.4.2 DigiTally codes

We now outline the stages required to complete a DigiTally transaction. We provide a simplified version of the protocol to emphasise the generation of codes required to complete a DigiTally transaction. The prototype implementation included the full protocol as discussed in Sections 5.4.3 and 5.5.1 to implement transaction chaining and group key scheme. Moreover, with regards to the length of codes, we chose to implement and evaluate 8-digit codes for the challenge ($Code_1$) and response ($Code_2$)[5].

There are two codes involved in completing a transaction. To simplify the discussion, we will assume that Alice is paying Bob.

[5]Refer to Section 5.4.3 for a discussion about the length of the codes.

1. $Code_1$: After the payee B (Bob) has entered into his SIM the transaction amount X and the phone number of the payer A (Alice)[6], his SIM generates a random nonce (N_B), and then computes a MAC on A, B, N_B, X and the log of previous transactions ℓ between the two parties[7]. This MAC and the nonce together make up the 8-digit $Code_1$, which is shown on the payee's device (Figure 6.2c). Alice similarly enters into her phone the amount X and Bob's number B[8]; it prompts her for $Code_1$, which she enters (Figure 6.2d). If the two parties agree on X, ℓ and each others' identity, then Alice's phone accepts $Code_1$. If there's a disagreement – whether due to attempted cheating or an honest mistake – her phone will generate an error[9].

2. $Code_2$: If $Code_1$ is correct, Alice's SIM card decrements her account's available balance by the transaction amount X and generates $Code_2$ to authenticate the transaction. $Code_2$ is also 8 digits long; it consists of a 4-digit nonce N_A generated by Alice's SIM and 4 digits from a MAC on A, B, X, ℓ and N_B. Alice then shows or tells $Code_2$ to Bob (Figure 6.2e). He enters $Code_2$ into his phone, and, if it is valid, his SIM increments his balance by X and a transaction log is displayed on his phone (Figure 6.2f). A similar transaction log is shown to Alice to confirm the completion of the transaction (decrementing her balance by X).

This is the simplest payment protocol we could devise that enables value to be transferred from Alice's card to Bob's by copying 8 digits in one direction and 8 digits in the other. Its security was analysed in the previous chapter. Our focus now is usability, and there is useful precedent from prepayment meters. The STS prepayment electricity meters widely used in Kenya transfer value using a 20-digit number, presented as five groups of four digits [8]. A householder buys codes from an ATM or sales agent and copies them into their electricity meter; codes can also be bought online, using M-Pesa. Thus, we know that our prospective customers can copy digits, and that even illiterate people can use a phone. The object of the study is to establish whether the DigiTally transaction flow is similarly usable.

Some care is necessary to ensure robustness in the face of mistakes. The DigiTally implementation includes various segments that execute atomic operations (enforced by the Java Card platform) to ensure that critical components of the payment protocol are fully executed (or reverted to their initial state if a problem occurs). This provides the ability to create reliable *checkpoints*. We checkpoint the transaction when $Code_1$ is successfully generated so that a transaction can be recovered if, for example, a merchant accidentally generates a second, new $Code_1$ before the customer replies to the first one. Additionally, as Alice's balance is decremented prior to displaying $Code_2$ on the payer's device, we have to ensure that if she is interrupted (e.g., by a flat battery), her SIM can still retrieve $Code_2$ later ($Code_2$ is saved in the final checkpoint of the transaction, and can be retrieved from the "last transaction" log). Checkpointing was also tested (see Section 6.7.2).

[6]In our trial, user identities are randomly generated numbers, each simulating a phone number.

[7]If no previous transactions exist, then the first transaction initialises the relationship between the two parties.

[8]Alice and Bob can pick each others' names from a menu. The first transaction stores the contact's details, and subsequent transaction can later retrieve information from the locally saved contacts (on the phone or SIM).

[9]For example, if Alice enters \$4 on her device, and Bob enters \$5, then Bob's $Code_1$ will generate an error on Alice's device. The same thing happens if the wrong phone number is selected.

DigiTally codes do not have to be kept secret, since they can be used only once and only by the payer and payee in a specific transaction. There is no added risk if other users observe the transaction or overhear the codes being exchanged verbally. This is in contrast to mobile payment systems that rely on secret codes that compromise the users' security if intercepted, and impose a burden on the payer to deliver them to the recipient out-of-band. Users of feature phones cannot use a secure messaging service, so probably have to use voice or SMS and hope for the best. As well as the (small) technical fraud risk, such codes can create anxiety and make dispute resolution problematic.

6.5 Method

The usability evaluation of DigiTally was conducted by the author of this thesis from the University of Cambridge (UK) and researchers from Strathmore University (Kenya), in early September 2016. Participants from Strathmore University (hereafter referred to as "the university") performed real-world transactions using DigiTally and then answered questions about their experience. As well as open-ended questions, we make use of the System Usability Scale (SUS), which is a standardised tool widely used for measuring usability. We provide the details below.

6.5.1 Participants

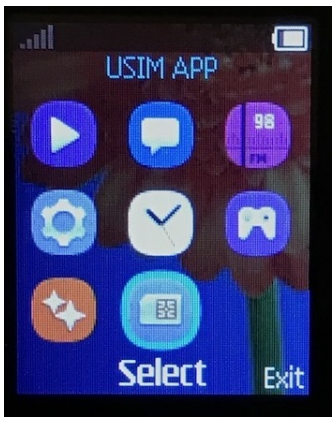
We wanted to test the technology as part of a pilot, and so we needed participants who could reveal weaknesses in our design. The aim was to maximise usability before introducing DigiTally in a larger-scale field test with target users. As an initial trial to establish any major issues with the DigiTally system itself, we recruited Kenyan participants from the university.

The participants were recruited following their registered interest in the study in response to an advertisement. Potential participants were interviewed with an aim to achieve diversity in terms of demographics, and to establish that participants would be able to give us detailed feedback to inform major re-designs before a field trial with a representative sample of rural users.

Twelve students and seven merchant staff were recruited as participants to use DigiTally for five days. Merchants were recruited from the university's cafeteria, bookshop, and a local coffee shop. According to Sauro [109], Vizri [118], and Nielsen and Landauer [97], very few participants are needed for early phase usability studies such as this, since adding more users tends to uncover the same issues, with "five" often referenced as the "magic number" of participants. Confidence intervals may be wide as a result, but the average SUS score should be stable [108].

For wider coverage, we included a greater variety of participants, which required larger numbers than the proposed "magic number 5". This is in line with more cautious estimates of 10-20 participants for a usability test (e.g., Faulkner [57] and Macefield [80]).

In total, we had 19 participants for this preliminary study. There were 7 female and 5 male students, who were studying a range of topics in different faculties, including finance, law, and information technology. As for the merchants, the cafeteria included 1 female and 2 male staff members; the bookshop had 1 female and 2 males; and the local coffee shop had 2 females (only one of whom was responsible for processing DigiTally transactions). Each merchant was given one feature phone to process DigiTally payments; staff members

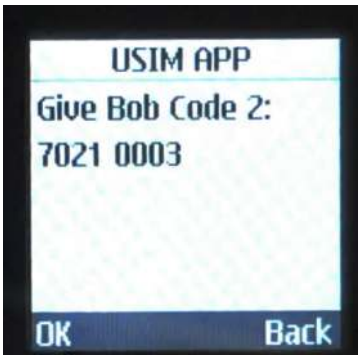


(a) Selecting DigiTally applet from the feature phone's application menu

(b) DigiTally menu shown on a feature phone

(c) Bob's phone displays $Code_1$

(d) Alice enters $Code_1$ into her phone



(e) Alice's phone displays the response code ($Code_2$), given to Bob to authorise the payment

(f) Bob's phone displays the transaction log after accepting $Code_2$ which finalises the transaction

Figure 6.2: DigiTally user interface.

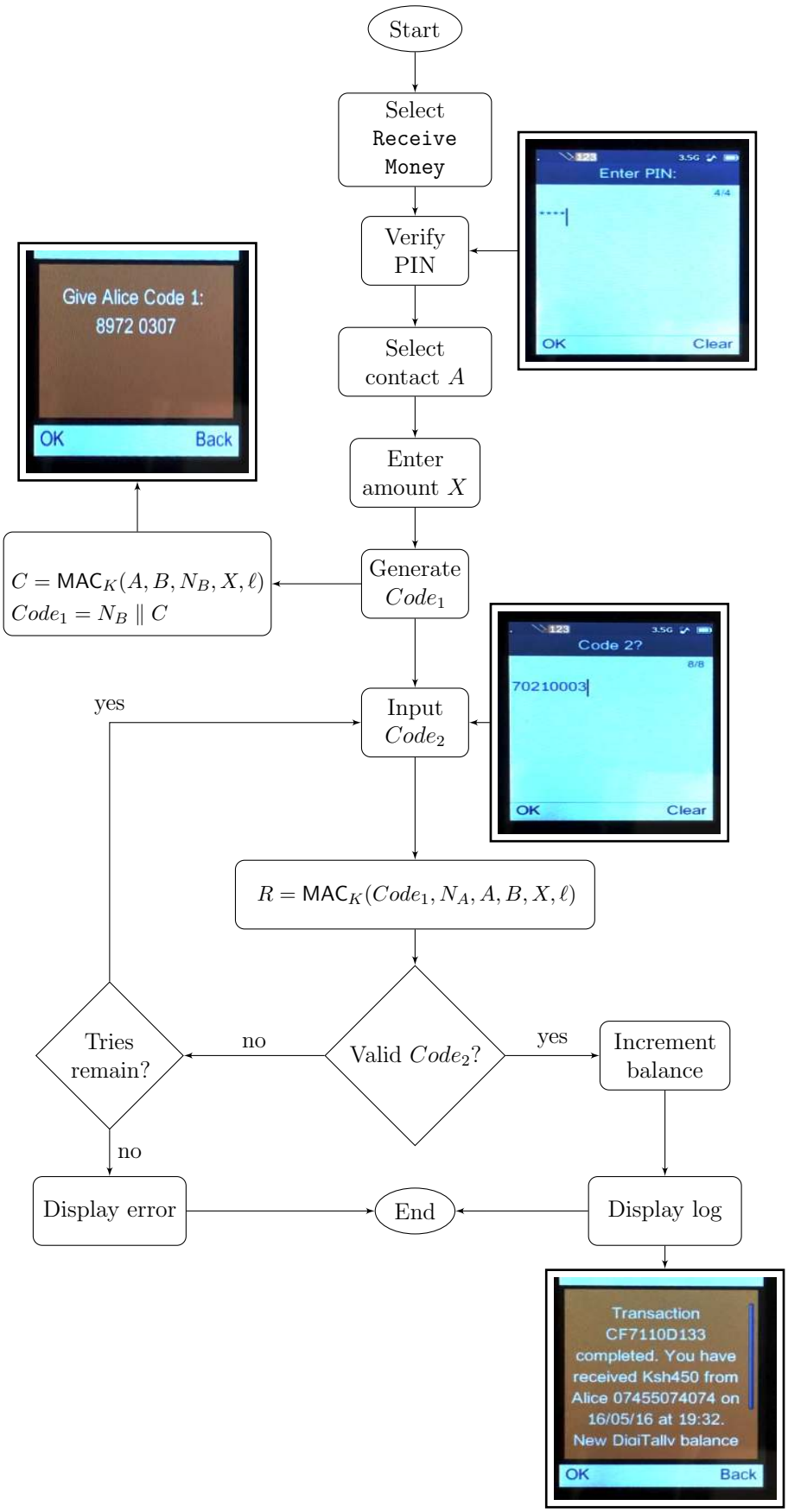


Figure 6.3: Recipient's (Bob B) transaction steps (this flow chart illustrates the simplified protocol; see Section 6.4.2 which discusses the prototype implementation).

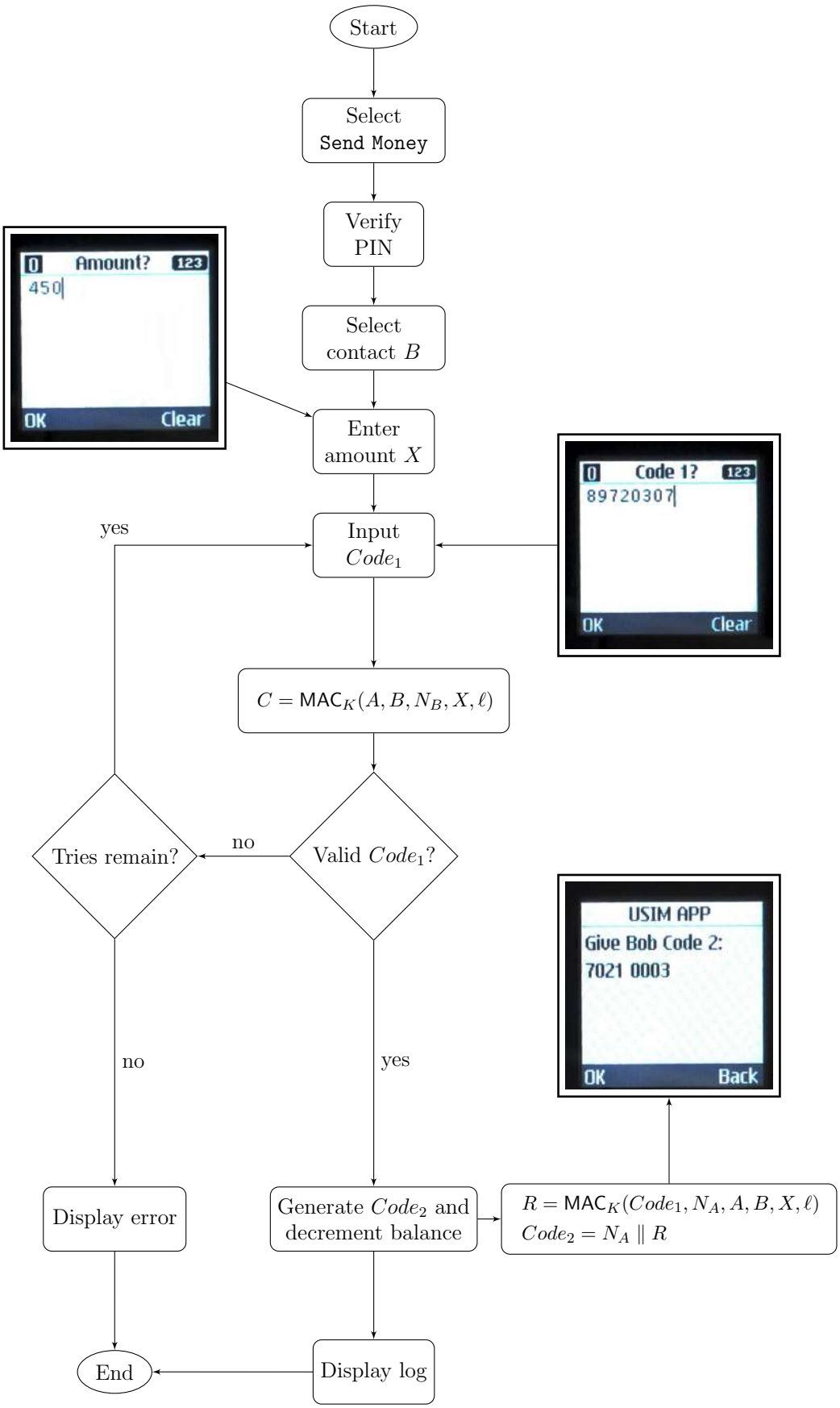


Figure 6.4: Payer's (Alice A) transaction steps (this flow chart illustrates the simplified protocol; see Section 6.4.2 which discusses the prototype implementation).

in the cafeteria and bookshop shared the device to process payments, while the coffee shop had one person operating the feature phone.

Because we did not provide cash-in and cash-out services (to convert physical currency to DigiTally balances, and vice versa), participants were motivated to use DigiTally to spend the balance they had in their SIMs.

6.5.2 Evaluation materials

Questionnaires and surveys are one of the most widely used methods for measuring attitudes [47]. They often involve asking participants to select one of a number of points on a rating (Likert) scale [111]. Good scales are valid (in that they represent the intended construct), and reliable (giving measurement consistency) [40, 111]. Standardised usability questionnaires and surveys are preferred because they are quantifiable, are economical to reuse, and allow for replicability of findings [111]. The most popular standardised survey for measuring attitudes towards usability is the SUS, favoured due to its brevity and being free to use [110]. The items in this survey factor into two sub-constructs of usability: system learnability (ease-of-use for new users), and system usability (defined as ease-of-use more generally) [110].

It was important to use feature phones for this study rather than the smartphones that many students normally use. Participants were thus given identical feature phones to test the basic usability of the technology first; a field test with representative users in rural Kenya who already use feature phones would be the next step.

We asked participants to complete a short pen and paper survey, which included the SUS, to make an initial assessment of DigiTally. Here, we asked participants for three free-text responses, to get insights into how they viewed DigiTally and where we might need to improve. These were:

1. What did you like about DigiTally?
2. What did you dislike about DigiTally?
3. Additional comments

6.5.3 Procedure

Students and on-campus merchants at the university were invited to try DigiTally and to provide their opinions of their experience of the system. This involved using dedicated feature phones to transact using the DigiTally applet, which was installed on each phone's SIM. The students' phones were preloaded with a DigiTally balance of Ksh 2000 (about $19.50) that could be used to make transactions with merchant participants who accepted DigiTally. The merchants (cafeteria, bookshop, and a local coffee shop) were given phones that had a zero balance on their DigiTally applet.

Before the trial, participants gave informed written consent and were shown how DigiTally works. After the trial, participants were asked if they were willing to complete a survey about their experience, which was entirely voluntary. This survey consisted of the SUS and the three open-ended questions outlined in Section 6.5.2 (Evaluation Materials).

Each day towards the end of business hours, the researchers visited each merchant to check their balances and reimburse them, in cash, the amount they had in their DigiTally

balance (after transferring the DigiTally balance to another phone, and thus resetting a merchant's balance to zero at the time of reimbursement). Each merchant signed a form indicating the date and amount of reimbursement. The reimbursement forms were necessary to create a detailed accounting of the money for our funders and auditors.

6.5.4 Ethics, data collection, and privacy

This study was approved by ethics committees at both universities involved in this research (one in the UK and the other in Kenya). To address the various concerns raised by the ethical review boards in both universities, we changed the design of the study to minimise any possible risks:

1. **Financial risks**: to address the risk of possible loss of personal funds, we provided the participants with a preloaded amount into their DigiTally applet (as explained in Section 6.5.3). We decided not to provide cash-in and cash-out services to eliminate the possibility that participants would use their own funds to increase their DigiTally balances.

2. **Privacy risks**: we were asked for assurance that the overlay SIM would not compromise the privacy of participants by interfering with the participants' regular SIMs. Although we initially intended to use an overlay SIM on top of a regular SIM, we decided instead to use the overlay as a regular SIM and provide users with dedicated devices. The reason for this decision is that *proving* that an overlay SIM is not compromising users' privacy was out of the scope of this project. Note that the overlay SIM (and the DigiTally applet) does *not* compromise users' privacy in our implementation, since it operates as an independent sandboxed applet and does not interfere with other applets' data if there are any installed.

Therefore, we consider the preliminary study to have no significant ethical implications. There were no known risks to participants (financial, psychological, emotional, or physical). Participants were required to provide informed consent before starting the study. The informed consent form outlined what the study involved from the participant, how long it would take, our commitment to confidentiality, and their rights as a participant. Participants were reminded, verbally and in the consent form, that their participation was entirely voluntary, and that they may choose to not answer particular or all questions and may withdraw from the study at any point without having to provide a reason, and without fear of penalty from the researchers.

This study did not involve deliberately deceiving participants in any way. This study also did not involve recruiting participants from vulnerable groups, such as children, patients, people with learning disabilities, people engaged in illegal activities, or people in custody. Participants were fully debriefed and their questions were answered throughout the study. Additionally, participants were given contact information for the lead researchers from both universities should they have any further enquiries.

The data we used for our evaluation includes the survey results, as well as the performance and transaction data processed by the SIM. By collecting these data, we were able to analyse performance and usability issues as we discuss in Results (Section 6.6). We collected the following:

1. Error rates (code-entry errors, and wrong PIN input)
2. Number of transactions
3. Number of attempts to unlock the SIM
4. Total amounts for all transactions (spent and received)
5. Transaction duration times

The SIM also stored the full log of the last transaction, which is overwritten when a new transaction is successfully processed[10]. This log contains $Code_2$ which can be retrieved after the transaction is completed. Recall that $Code_2$ can be used only once and with the right recipient (the identity of the recipient is one of the inputs required to generate the code), but a user might want to retrieve $Code_2$ in case their battery goes flat before exchanging the code with the recipient.

The collected data were stored in the SIM card (a trusted tamper-resistant element). DigiTally does not leak any information outside this trusted element. Therefore, performance data are inaccessible without the correct authenticator. Only pre-configured devices, programmed to provide the correct authenticator, had access to the SIMs to obtain the performance measurements. Each SIM was programmed to lock itself after a certain number of failed attempts to access the contents of the trusted tamper-resistant environment. At the completion of the preliminary study, we retrieved the phones from all the participants to extract the performance data from the SIMs and save it for later analysis. All the stored data were encrypted after extraction.

With the exception of participant names, no Personally Identifiable Information (PII) were collected in the study (such as date of birth, national ID numbers, etc.), as they were irrelevant to the scope of the study. We carefully explained to all participants what data were being collected, by demonstrating many full transactions on two phones and explaining the performance variables to be measured for the purposes of the preliminary study.

6.6 Results

The following sections report on the actual error rates and transaction times, as well as the SUS results. We present descriptive statistics to help summarise the data and to show any emerging patterns. We make no claims about statistical significance, which would be reserved for inferential statistics. This would not be appropriate since the tested population do not represent the intended users of DigiTally, and were recruited in order to reveal any major usability issues with the technology itself. We thus make no claims about generalisability of the results.

We also discuss the open-ended results and the lessons learned through the feedback obtained from the users. We discuss the features that participants considered to be the most important, and indicate issues that may need to be re-evaluated to improve usability.

Note that there is a difference between the total number of transactions completed by students, and the total number of transactions completed by merchants. This is due to

[10]This log is informational only, and is different from the cryptographic parameters securely stored on the SIM and used in the payment protocol as discussed in Section 6.4.2. Refer to Chapter 5 for more details.

	No. of transactions	PIN errors	$Code_1$ errors	$Code_2$ errors	Total code errors	Average time (seconds)
S1	30	2	0	8 (26.7%)	8 (26.7%)	30.9
S2	28	3	9 (32.1%)	0	9 (32.1%)	24.4
S3	18	3	2 (11.1%)	8 (44.4%)	10 (55.6%)	28.1
S4	22	0	9 (40.9%)	1 (4.6%)	10 (45.5%)	44.9
S5	29	1	1 (3.5%)	0	1 (3.5%)	24.2
S6	26	1	0	1 (3.9%)	1 (3.9%)	54.3
S7	26	0	1 (3.9%)	0	1 (3.9%)	50.9
S8	28	0	5 (17.9%)	0	5 (17.9%)	32.4
S9	10	0	0	4 (40.0%)	4 (40.0%)	28.8
S10	29	1	0	0	0	37.1
S11	8	2	0	0	0	42.1
S12	22	0	5 (22.7%)	0	5 (22.7%)	38.9

Table 6.1: The frequency and types of errors made by student participants and the percentage of each participant's transactions that their errors affected

	No. of transactions	PIN errors	$Code_1$ errors	$Code_2$ errors	Total code errors	Average time (seconds)
M1	58	2	0	4 (6.8%)	4 (6.8%)	40
M2	61	0	0	6 (9.8%)	6 (9.8%)	43.9
M3	58	2	0	4 (6.8%)	4 (6.8%)	69.8

Table 6.2: The frequency and types of errors made by merchant participants and the percentage of each participant's transactions that their errors affected

the fact that students moved funds between their phones, perhaps to test the system and in some cases to demonstrate it to their friends; moving funds between devices back and forth did not incur any costs in terms of transaction fees.

6.6.1 Errors and speed

Students completed an average of 23 transactions. The highest number of transactions completed by one participant was 30; the lowest was 8. For merchants, the average number of transactions was 59. The highest number of transactions completed by one merchant was 61; the lowest was 58. Tables 6.1 and 6.2 display a summary of the *Error Rates* and *Time on Task*, which should be considered against the number of transactions each participant successfully completed.

6.6.1.1 Error rates

DigiTally captured the number of errors participants made while trying to complete the task. Student participants made the most errors when entering the first code ($Code_1$) presented to them by the merchant. The number of errors here ranged from 0 to 9, with 7 of the 12 student participants making this error. These were non-critical errors since they do not prevent successful completion of the transaction. However, future trials will need to test the ease with which users can recover from this and other errors. For $Code_2$, the number of errors ranged from 0 to 8, with 5 of the 12 student participants making this error.

The merchant participants made the most errors when entering $Code_2$ which is required to authorise and complete the transaction. The number of errors here ranged from 4 to 6. Merchant $M2$'s $Code_2$ errors were higher because they included errors experienced during training. We adjusted the number in the table to reflect the errors experienced during the study without those experienced during the training. Merchants made no $Code_1$ errors; this type of error would occur if they used DigiTally to make payments instead of receiving

(the merchants made a single daily payment using DigiTally for the reimbursement of transactions, see Section 6.5.3).

We include PIN errors in Tables 6.1 and 6.2 for completeness. However, we do not consider entering PINs to be a major problem, at least no more than in any other system that uses them. For clarity, we report (in brackets) the proportion of errors relative to the number of transactions completed as a percentage (for non-zero values).

6.6.1.2 Time on task

DigiTally recorded the time on task for each participant. The transaction timer starts when a menu option is selected (`Send Money` or `Receive Money`), and the timer is stopped before the transaction log is displayed to the user (the transaction log includes the transaction's duration time).

The average time for students to complete a transaction was 36.4 seconds. Their average completion times ranged from 24.2 seconds to 54.3 seconds. It is worth noting that a large number of errors by a participant did not necessarily translate into longer average time spent on transactions. For the merchant participants, the average time to complete a transaction was 51.23 seconds. Their average completion times ranged from 40 to 69.8 seconds. Merchants' transaction times are larger since a merchant can start a transaction, give $Code_1$ to the payer, then complete a few tasks to serve the customer until $Code_2$ is entered. Such tasks include having to prepare food at the same time as processing DigiTally payments, which is the case for merchant $M3$. For merchants dedicated to processing payments (cashier roles), the transaction times are lower ($M1$ and $M2$).

6.6.2 SUS results

The SUS is not diagnostic: it will not reveal specific problems, but it does give an idea about overall *ease-of-use*, and whether significant changes might be needed.

To give a clear idea of the results, the SUS is calculated so as to provide a score out of 100. However, the SUS Score is not a percentage. A score of 68 actually falls at the 50^{th} percentile (i.e., the average SUS score is 68). If the score is below 68, there are likely to be serious usability problems that need tackling. A score of 80.3 or higher is ideal.

The average SUS score for DigiTally was 78.8, which is considered "Good", and would be given a "B+" grade. The lowest score was 50, which is considered "Poor", and is equivalent to a "F" grade. This score was given by a merchant and was the only score considered "Poor" by SUS standards. The highest score was 100, which is the best possible, and is equivalent to an "A+" grade. Eight participants gave the equivalent of an "A+" grade.

For merchant participants, the average SUS score was 71.4, which is considered "Good" and equivalent to a "C+" grade; for student participants, the average SUS score was 83.1, which falls just short of being considered "Excellent", and is equivalent to an "A" grade.

Although the SUS is intended to be a measure of ease-of-use, Lewis and Sauro argue that it can also be used as a measure of *learnability* (using items 4 and 10 of the SUS) [79]. As with calculating the SUS score, learnability can be calculated to give a score ranging from 0-100. The average learnability for this initial trial was 82.9, which falls just short of "Excellent" and would be given an "A" grade. The remaining 8 items are what Lewis and Sauro call a measure of *usability* [79]. For the current initial trial, this score was 77.8, which is also considered "Good" and would be given a "B+" grade.

6.6.3 Responses to open-ended questions

Participants' answers to the three open-ended questions were categorised into several themes. This qualitative data analysis involved two researchers independently coding user comments to identify common themes, supported by quotations. These researchers then came together to assess agreement and categorised themes based on a well-established definition of usability (ISO-9241), which consists of effectiveness (usefulness), efficiency (ease-of-use), and satisfaction.

6.6.3.1 Perceived usefulness

1. **Money saving**. Most participants mentioned the benefit of there being no transaction fees. They pointed out that this would be useful for those in poor communities, as well as being attractive to price-sensitive customers and merchants. Aside from helping users avoid transaction fees, one participant mentioned that it also meant they did not have to use a smartphone, making DigiTally even more cost-friendly.

2. **Network independent transactions (interoperability)**. Many mentioned the benefit of not having to rely on network coverage. For some, this made the DigiTally transaction process seem more reliable. It also makes it more predictable as users do not have to wait for a confirmation SMS, which with M-Pesa can take up to 30 seconds.

3. **Security**. The general consensus was that DigiTally seemed very secure. The codes were a major factor behind this perception. While recognising this benefit, a few participants suggested that the codes were too long, and recommended that the developers consider shortening them.

4. **Money tracking**. Participants liked being able to review their last transaction and balance. For some, however, this was not enough. It was suggested that this feature would be more useful if the user could review all previous transactions.

6.6.3.2 Perceived ease-of-use

1. **Ease**. Participants perceived DigiTally to be simple and easy to use. Some clarified that DigiTally was easy to use *after* a learning period. One merchant described the process as cumbersome; this merchant was both serving customers and taking payments. Other merchant participants were either less busy or were dedicated cashiers. One participant also stated that DigiTally might be harder to use for the elderly and those with poor eyesight.

2. **Learnability**. DigiTally was most often praised for its learnability; in general, participants felt like DigiTally was easy for a first-time user to understand. Even the busy merchant ($M3$) who had complained that the system was cumbersome said she had no difficulty training a staff member to use it. The same merchant also warned that learnability might be lower for some, including the target population, where there might be more illiteracy and less education more generally. Another merchant was curious to know how money would be deposited in a production DigiTally system (this merchant is already an M-Pesa agent).

3. **Speed**. Participants found transacting with DigiTally to be relatively fast, once they knew the process. Two out of the eight merchants said that DigiTally had too many steps and was time-consuming compared to cash or M-Pesa. Requiring as much effort from the merchant as from the customer to give and receive codes was problematic when the merchant had a lot of customers and needed to perform other tasks at the same time. One merchant also did not like having to search for the customer in their contacts list. One participant pointed out that the speed of transaction would be less of an issue in rural areas.

4. **Errors and recoverability**. One participant stated that it is hard to make errors that would cause the user to lose money, and that they found errors easy to rectify. Many others did not agree. In general, error reset was considered too difficult. Although the process for error recovery was perceived as cumbersome, none of the users had to go through it during the trial. See Section 6.7.2 for more details.

5. **Cashlessness**. Two participants noted that by using DigiTally they would not need to carry cash around. This was considered convenient. It was also noted that merchants do not have to find the exact change when dealing with customers using DigiTally. This benefit is shared with other electronic payment systems; Kenyans already like M-Pesa as it dispenses with the risk and the inconvenience of cash.

6. **Codes**. Although many did not have much of a problem with the codes, finding them short enough and necessary for security, some stated that they did not like them, mostly because the code or the process was too long or awkward. Having to exchange and input codes was also perceived as an opportunity to make errors. On the other hand, one of DigiTally's perceived advantages was the deterministic nature of the transactions, because exchanging the codes provided immediate feedback that a transaction was completed, without needing to wait for an SMS.

7. **Distance**. Sending money to a remote payee was identified as a potential problem. It was pointed out that errors could be more likely and recovering from them would be harder if users were trying to complete a transaction at a distance.

6.6.3.3 Satisfaction

1. **Likeability**. Participants liked using DigiTally, using words like "happy", "good", "best", "seamless", "enjoy", "enthusiasm", "encouraging", "smart", and "satisfied" to describe their experience. Comments suggest that likability could be improved by targeting issues associated with the length of the codes and with recovering from errors.

2. **Other services**. In addition to a merchant who wanted to know how to do cash-in and cash-out transactions, one student participant also indicated that they would like to be able to do this using DigiTally. We did not provide cash-in and cash-out services during the trial.

6.7 Observations

We describe in the following sections our observations during the preliminary study. These observations were documented during regular visits to merchants to answer their questions, and during daily visits to reimburse merchants for their DigiTally transactions (as discussed in Section 6.5.3). We highlight two important observations, visual cues and the error-recovery process, and discuss the lessons learned.

6.7.1 Visual cues

We observed that participants chose to display the codes so that the other participant could see them and enter them into their own device. None of the participants indicated that they exchanged the codes verbally, and some stressed that visual exchange is easier. In crowded areas, a verbal exchange of the codes could lead to misheard digits. The cafeteria is a crowded environment with long lines of customers, where a window separates the cashier and the customer, and a small opening at the bottom of the window allows exchanging cash or passing cards (Figure 6.5). In this environment, we observed multiple DigiTally transactions where the only verbal exchange was to acknowledge that a transaction was completed by saying "*it's OK*", and in some cases just a nod. When we mentioned this observation, most participants agreed that "*it's much easier*", and gave the example of the cafeteria. In general, once participants had some experience with the system, we noticed that they would always show the code and never speak it even in quiet areas. In the transaction flow people developed, visual cues were used to complete each major step, starting with the customer declaring that they want to use DigiTally by showing a yellow label on the back of the phone that identifies the participant by name (Figure 6.6). However, we should not neglect the fact that some users in the target population for financial inclusion might be visually impaired.

6.7.2 Error recovery

We designed error recovery to prompt the user for three dummy inputs (any sequence of digits) to ensure that they were fully aware that they were indeed resetting the saved transaction data; in our implementation, a wrong reset could lead to inconsistent states on the SIMs involved. Inconsistent states currently require a hard reset of the relationship between the two users by deleting the contact information and creating a new contact on each phone.

Before each prompt for $Code_2$, the recipient is shown a confirmation of the transaction details. After the dummy inputs are provided, the session is reset by discarding $Code_1$ as well as any intermediate results that were saved for validating $Code_2$. Now the recipient (merchant) can start a fresh transaction: the next time a transaction is initiated between the same two parties, a new $Code_1$ is generated. Even with default-text entry enabled in the applet to fill in the previously entered code, participants requested an easier way to rectify a mistake.

As error recovery was performed only by one student participant during the actual trial, most were likely evaluating the training they received rather than anything they actually did. Two users needed to reset a transaction during training. One participant commented: "*For the error issue: I think you should make it simpler to correct the error and provide instructions on how to correct it once the error is made since you will not be*

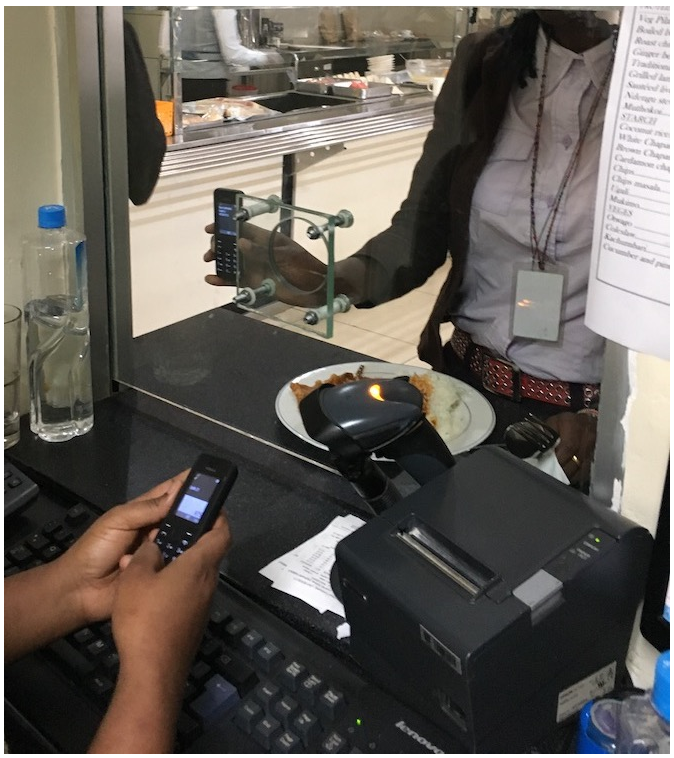


Figure 6.5: The cafeteria cashier (recipient) using the phone on the left, while a student participant (payer) displays $Code_2$ through the window to authorise the payment

Figure 6.6: Participants displaying the back of their phones (yellow DigiTally label shown on the back)

around to show them. For example, when an error of different amounts that needs one to put code 1 three times, I think you should write 'repeat code 1 three times to correct the error'." The other participant agreed: "*If there's a simpler way of resetting a wrongly transacted code DigiTally will be better.*" None of the other participants, including the merchants, had to use this error-recovery method, likely due to the training component instructing users to agree on the amount before proceeding with the transaction. We emphasised the importance of this step; perhaps the burden of going through the error recovery process helped motivate careful exchange of the codes.

Our prototype error-recovery mechanism needs a redesign. A simple alternative would be a menu option to reset transaction data, requiring authentication with the user's PIN rather than requiring them to enter dummy inputs.

6.8 Discussion

This chapter reports the results of a preliminary study with an early prototype of DigiTally, tested on participants from a university in Nairobi. We described how DigiTally involves sharing two 8-digit codes to protect users from unauthorised payments while at the same time allowing them freedom from reliance on network coverage. This has an advantage that the payment service operator has no marginal transaction costs and can offer zero fees for some transactions. However, the code-sharing process has always presented a possible usability challenge. To test this and other aspects of this early prototype, DigiTally was introduced to Kenyan participants in order to identify and resolve any major usability issues before a later field trial with a more representative sample of service users.

The errors made in sharing codes suggest a need to make recovery from this type of error more intuitive. Nevertheless, most participants completed the task with few code-sharing errors. The average speed for every student was less than a minute, and overall the average transaction speed was close to half a minute. Given our observations of participants and the comments they made, it seems that the process, overall, seemed straightforward and fast. The SUS scale gives insight into the extent to which DigiTally's usability might inhibit its use. One of the lessons learned was the need to demonstrate clearly to users: (a) how errors can be avoided; and (b) how the codes prevent cheating.

Observations with initial trial users and open-ended answers following the trial indicate that users' main fear is the difficulty of recovery from errors. This is already an issue for M-Pesa when people send payments to the wrong phone number (it is a well-known problem, and one of the project researchers experienced it first-hand). Recoverability is especially important in the initial trials and adoption phase of a new system because expert help will not be available in remote villages; users' ability to figure out how to recover from mistakes may well be the difference between adopting and rejecting DigiTally.

Higher SUS scores tend to predict loyalty and word-of-mouth recommendations [109]. Users with scores over 80 (cf. the average score of 83.1 for student participants) are called "Promoters" because they are more likely to recommend a system, while users with a score below 60, called "Detractors", are more likely to say negative things about a system. DigiTally is in a position to create "Promoters", especially for student participants.

There are various things we can do to make DigiTally more usable for busy merchants. For example, we can give merchants a smartphone app that reads the customer's phone number from a QR code on a sticker on their phone, display $Code_1$, and read $Code_2$ from their phone screen. This way the merchant does no more work than with M-Pesa. Two of

the merchants preferred DigiTally to M-Pesa because of speed; DigiTally does not force a cashier to wait for up to half a minute while a payment confirmation makes its way through Nairobi's congested mobile network. DigiTally takes more keystrokes, but the outcome is then immediate.

The fact that DigiTally has zero marginal costs also means that it can be offered as a zero-fee payment mechanism between friends and family, like personal cheques in the UK. Loan clubs, such as Rotating Savings and Credit Associations (ROSCAs) or savings clubs, and money-guards (used to enforce savings through the commitment of funds), whether formal or informal, are an important part of the financial ecosystem in many less developed countries. See Collins *et al.* [36] and Banerjee and Duflo [15] for more information about financial inclusion and the financial tools used by the poorest demographics (living on less than $2 a day).

Given our relatively high SUS, learnability, and usability scores, the challenge is encouraging first time use. The factor most apparently affecting first time use, based on free-text responses, was the 8-digit codes. The most frequent suggestion was to remove or reduce them. Although many participants were familiar with M-Pesa, and thus sharing an 11-digit phone number, this is potentially less time-consuming and error-prone because it is the same number every time for the recipient (their own phone number) and a single time entry for the sender. It also requires no input from the recipient, so a merchant can focus on other tasks.

Participants perceived DigiTally to be secure, and the codes were the reason behind this perception. As mentioned in Section 6.6.3.1 item 3, participants asked if these codes can be shortened. However, this would not be possible without compromising the security of the system (as discussed in Chapter 5). A few participants also requested a better money tracking tool: some participants liked that statistics about their transactions were available (e.g., amounts sent and received as explained in Section 6.5.4), and some users requested the ability to view more than just the last transaction (as discussed in Section 6.6.3.1 item 4). This is not a critical issue, as we can engineer the system to include more transactions as allowed by hardware constraints and storage capacity.

Because migrant remittances are a key application of mobile payment systems (such as M-Pesa), some of our participants discussed the possibilities of remote transactions (see Section 6.6.3.1 item 4). We designed DigiTally primarily for face-to-face offline transactions and not for remote transactions that would require a medium to exchange the codes. However, if such a medium exists, then DigiTally codes can be exchanged online to process remote transactions. Codes can be exchanged using SMS or over the phone; they could even be exchanged by post. DigiTally can be viewed as a platform to do the cryptographic operations required to process payments, on users' phones rather than on a centralised server, and systems can be built on top of DigiTally to perform remote transactions – relying on the DigiTally SIM to do the cryptography.

DigiTally was praised for being quick, easy to use, and easy to learn. Kenyans are already familiar with M-Pesa, which DigiTally was designed to mimic; that may have helped directly, while other factors our participants liked, such as being able to review transactions, may have been features that they would like to see in M-Pesa. Therefore, the results in this chapter do not measure how DigiTally would be perceived by users who have never used any mobile payment system.

But M-Pesa, like most mobile payment systems, cannot work offline, and therefore fails to provide service to the poorest communities. Just as M-Pesa appealed most strongly to

people with phones but no payment cards, so also DigiTally should appeal most strongly to people with no network service at all.

This was confirmed when we made two field trips to scout possible sites for a second-round field trial. When we visited a small town near Nairobi with good network coverage, stakeholders were interested in playing with the system, but saw its main benefit as being potentially programmable so that it could support their specific applications. When we visited Busia, a rural community near Lake Victoria with very poor network coverage, stakeholders were delighted, and played with it for hours. They considered DigiTally to be just what they needed to solve their problems with network coverage and reduce transaction costs.

6.9 Limitations

In this section, we discuss various limitations and possible future work for offline payments in general and DigiTally in particular[11].

1. **Loss of device**: loss of the device, and the SIM it includes, may lead to loss of funds. The dispute resolution and consumer protection mechanism we envision for DigiTally includes refunding the customer with the last known balance, after processing received transactions from other customers if any are available (that can effect the last known balance, and provide proof that the customer's balance must be either increased or decreased based on recent activity). However, this may create an opportunity for fraudulent claims which we discuss below.

2. **Fraud**: a user can claim loss of a device to claim back funds even though those funds were already spent. However, we envision that this kind of fraud can be rate-limited; a simple verification to check how many times a user claims loss of a device can reject the user's application for excessive claims and terminate the user's account (prevent the user from registering as a client in the future to prevent fraudulent claims; this can be an effective deterrent). Moreover, social factors may impose self-regulation to avoid negative reputation within a community.

3. **Fragility of the overlay SIM**: the adhesive on the overlay SIM we used (as provided by Taisys) can wear off if the overlay is regularly removed and reapplied in the device. However, the overlay SIM is 1) used solely for prototyping, and a production system would likely embed the DigiTally code in a regular SIM, which requires the cooperation of the SIM provider, and 2) as we designed the prototype, we envisioned that the overlay SIM would be applied correctly once and there is no reason to remove and reapply the overlay SIM. We mention the fragility of the overlay SIM as a possible limitation for the following reason: during our research visit to Kenya, we were informed that the leading MNO instructs their customers to remove the overlay SIM, provided by the MNO's competitors, if that customer wants to proceed with in-shop customer service. If the customer does not remove the overlay, then they are denied service. This creates a venue for the leading MNO to undermine its competitors by forcing customers to comply (and would ultimately

[11]This section is not included in our paper [20]. The content is relevant to a large-scale deployment, and some of the issues discussed were raised after project completion.

lead to customers choosing not to utilise the services of the MNO's competitors). However, we stress that this problem is vendor-specific: Taisys, or another vendor, may introduce to the market a more robust overlay SIM that can be easily and regularly removed and reapplied.

6.10 Conclusion

We designed and developed an early prototype of DigiTally, an offline phone payment system, and tested it on Kenyan participants in a preliminary study. In addition to error rates, transaction speed data, and SUS scores, we reported on supporting data from demonstration sessions (observed behaviours and comments), and free-text written responses at the end of the study. Our results indicate that participants found DigiTally easy to use and that they liked key aspects of the system, including its perceived security and that it did not require network coverage to process payments.

We demonstrated that DigiTally can be used for making payments without network coverage or transaction fees. Some specific technical improvements are needed, most notably the process of recovery from errors. We discovered that while DigiTally is slightly less convenient to use than existing mobile payment systems, the added burden is not excessive; people in areas with poor network coverage are eager to use it. Furthermore, the burden mostly falls on merchants, and busy merchants are likely to have enough money to buy better terminals.

There is a realistic prospect of developing DigiTally into a workable system that will extend mobile payments to the millions of people who are currently excluded from the world of electronic payments. However, this was not an option in Kenya because of a change in the business environment. Just after we completed the field trial, Airtel decided to minimise their mobile payment services in Kenya turning the M-Pesa operator, Safaricom, into the dominant player; and Equity Bank, with whom we had had discussions, stopped issuing overlay SIM cards to its customers. That left Safaricom as the only feasible deployment route, and they were not interested. In the absence of competition, they say their way forward is in marketing gambling apps to existing customers rather than in extending service to the rural poor.

Chapter 7

Conclusion

The core theme of our research was creating more resilient payment systems. To this end, we investigated several issues related to security engineering in general, and payment systems in particular. We highlighted resilience features in existing payment systems that provide network and system resilience features (through offline capabilities and dedicated networks), usability features (e.g., familiarity and simplicity), dedicated tamper-resistant hardware (to prevent loss of critical data and tampering with the protocol implementation), censorship-resistance features included in decentralised systems, low barriers of entry that can lower the friction required for participation, and decentralised transaction processing to alleviate the performance bottleneck resulting from central transaction processing.

We highlighted the lack of trust and reputation metrics in open-membership systems; the lack of such metrics exposes such systems to various network attacks. We investigated how using a payment system, with a pseudo-currency, creates useful metrics for building trust and reputation systems. We aimed to incentivise correct behaviour, to generate metrics to identify which nodes are most interconnected, and empower nodes to shift trust to other nodes. In other words, we facilitate the mechanisms required to democratise trust and power, by empowering participating nodes to vote for nodes that deserve their trust. We consider this a key building block to enable cooperation in open-membership systems that cannot prevent malicious activity through centralised operators.

We reviewed performance bottlenecks in Bitcoin that can cripple system performance if the network is exposed to spam campaigns. Our goal was to understand performance and resilience in cryptocurrencies. Although these are decentralised by design (at least theoretically), there are failure points that require more research. We were the first to highlight how a spam campaign on Bitcoin could degrade system performance leading to higher processing delays at the cost of a fairly modest sum in transaction fees. A large amount of spam can increase the backlog of unconfirmed transactions, leading to delays. To cope, senders pay higher fees, and so does the attacker – but the attacker also leverages the higher fees paid by others. We found that over 20% of Bitcoin transactions were spam during the 10-day period at the peak of the campaign, which increased delays in processing non-spam transactions from 0.33 to 2.67 hours on average. The attack cost about $49,000 to mount. The stated motivation behind the stress test campaign was to provide a justification for raising the Bitcoin block size limit. This was achieved in due course with a Bitcoin fork. Adversaries that are richer and more determined might disrupt the network more, and for longer than 10 days. The experience of this attack gives some idea what might be expected. This leads to the unfortunate result that Bitcoin is losing its initial appeal for low-value transactions and micropayments, since transaction fees are

currently higher and less competitive compared to traditional, and more familiar, payments systems. In addition, Bitcoin uses substantial amounts of energy, yet alternatives for decentralised consensus are unable to completely prevent attacks. We are not convinced that it is the way forward for low-value payments.

We therefore built on our experience to create an offline payment system, *DigiTally*, to extend financial inclusion efforts in developing countries. Offline payments are already used for electricity (both in prepayment meters and pay-as-you-go solar), we aimed to extend them into a general-purpose payment system, to increase service resilience in the face of network congestion or outage, and provide service to currently excluded areas. The goal of DigiTally was to tackle the network coverage and reliability problems by processing payments offline reliably and deterministically. We aimed to create these capabilities while simplifying the user interface and prioritising for usability, simplicity and speed, in order to avoid usability problems that can be a hindrance for uptake.

We analysed and discussed the offline payment protocol we created for DigiTally. The main motivation for using short message authentication protocols was to construct a lightweight protocol that can be used offline to finalise payments, while prioritising for resource-constrained devices and maintaining an acceptable level of usability. We discussed the limitation of verifying such protocols using BAN logic. We stress that protocol designers need to be good curators of security state, and also pay attention to the interaction between online and offline functions. We designed the protocol to preserve the integrity of transactions, by creating short authentication codes that can be exchanged between participants to get better tradeoffs between security and usability; our aim was to develop usable human-friendly protocols that can be used in constrained offline environments. We aimed to maximise usability in off-network transactions by minimising the number of digits a user has to speak, hear and type, while providing robust recovery mechanisms for the inevitable errors and making sure there isn't any scalable attack that is large enough to care about.

We then developed a prototype system, which lets users make offline payments by copying short strings of digits from one mobile handset to another. DigiTally involves sharing two 8-digit codes to protect users from unauthorised payments while freeing them from reliance on network coverage. The payment service operator will typically have no marginal transaction costs and can offer zero fees for some transactions. One of DigiTally's perceived advantages was the deterministic nature of the transactions; exchanging the codes provided immediate feedback that a transaction was completed.

In addition to error rates, transaction speed data, and SUS scores, we reported on supporting data from demonstration sessions (observed behaviours and comments) and free-text written responses at the end of the study. We discussed the lessons learned from our field visits and initial usability evaluation conducted in Nairobi, Kenya. We found that, although offline payments involve copying codes in both directions between the payer's phone and the payee's, the extra workload was acceptable to most users.

More research is needed to integrate online payments, offline payments, and delay-tolerant networking in general. The value of this research is not just to extend and enhance payment systems, but also all sorts of other electronically enabled services, from pay-as-you-go solar panels to agricultural subsidies and payments from aid donors.

We hope that this thesis is helpful for researchers and policy makers interested in creating resilient payment systems, and that our research and fieldwork can inform and assist future financial inclusion efforts.

Bibliography

[1] Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the economics of anonymity. In *International Conference on Financial Cryptography and Data Security*, pages 84–102. Springer, 2003.

[2] Carlisle Adams. *Have money, will travel: A brief survey of the mobile payments landscape*. Office of the Privacy Commissioner of Canada, 2013.

[3] Ross Anderson. UEPS–a second generation electronic wallet. In *European Symposium on Research in Computer Security*, pages 411–418. Springer, 1992.

[4] Ross Anderson. Why cryptosystems fail. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 215–227. ACM, 1993.

[5] Ross Anderson. *Security Engineering – A guide to building dependable distributed systems*. John Wiley & Sons, 2008.

[6] Ross Anderson. Risk and privacy implications of consumer payment innovation. In *"Consumer Payment Innovation in the Connected Age"*, pages 29–30, 2012.

[7] Ross Anderson and Khaled Baqer. Reconciling multiple objectives–politics or markets? In *International Workshop on Security Protocols*, pages 144–156. Springer, 2017.

[8] Ross Anderson and Johann Bezuidenhout. Cryptographic credit control in pre-payment metering systems. In *IEEE Symposium on Security and Privacy (SP)*, pages 15–23. IEEE, 1995.

[9] Ross Anderson, Charalampos Manifavas, and Chris Sutherland. NetCard–A practical electronic-cash system. In *International Workshop on Security Protocols*, pages 49–57. Springer, 1997.

[10] Elli Androulaki, Mariana Raykova, Shreyas Srivatsan, Angelos Stavrou, and Steven M. Bellovin. PAR: Payment for anonymous routing. In *Privacy Enhancing Technologies Symposium*, pages 219–236. Springer, 2008.

[11] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.

[12] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking Bitcoin: Routing attacks on cryptocurrencies. In *IEEE Symposium on Security and Privacy (SP)*, pages 375–392. IEEE, 2017.

[13] Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On Bitcoin and red balloons. In *Proceedings of the 13th ACM conference on electronic commerce*, pages 56–73. ACM, 2012.

[14] Adam Back. Hashcash–A denial of service counter-measure, 2002.

[15] Abhijit Banerjee and Esther Duflo. *Poor economics: A radical rethinking of the way to fight global poverty*. Public Affairs, 2011.

[16] Khaled Baqer and Ross Anderson. Do you believe in Tinker Bell? The social externalities of trust. In *International Workshop on Security Protocols*, pages 224–236. Springer, 2015.

[17] Khaled Baqer and Ron Steinfeld. Distributed payment systems. 2014.

[18] Khaled Baqer, Johann Bezuidenhoudt, Ross Anderson, and Markus Kuhn. SMAPs: Short Message Authentication Protocols. In *International Workshop on Security Protocols*, pages 119–132. Springer, 2016.

[19] Khaled Baqer, Danny Yuxing Huang, Damon McCoy, and Nicholas Weaver. Stressing out: Bitcoin "stress testing". In *Third Workshop on Bitcoin and Blockchain Research, affiliated with the International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2016.

[20] Khaled Baqer, Ross Anderson, Jeunese Adrienne Payne, Lorna Mutegi, and Joseph Sevilla. Digitally: Piloting offline payments for phones. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 131–143, Santa Clara, CA, 2017. USENIX Association. ISBN 978-1-931971-39-3. URL `https://www.usenix.org/conference/soups2017/technical-sessions/presentation/baqer`.

[21] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. Bitter to better–how to make Bitcoin a better currency. In *International Conference on Financial Cryptography and Data Security*, pages 399–414. Springer, 2012.

[22] Jörg Becker, Dominic Breuker, Tobias Heide, Justus Holler, Hans Peter Rauer, and Rainer Böhme. Can we afford integrity by proof-of-work? Scenarios inspired by the Bitcoin currency. In *The Economics of Information Security and Privacy*, pages 135–156. Springer, 2013.

[23] Alex Biryukov and Ivan Pustogarov. Proof-of-work as anonymous micropayment: Rewarding a Tor relay. In *International Conference on Financial Cryptography and Data Security*, pages 445–455. Springer, 2015.

[24] John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In *Cryptographers' Track at the RSA Conference*, pages 114–130. Springer, 2002.

[25] Bruno Blanchet. CryptoVerif: Computationally sound mechanized prover for cryptographic protocols. In *Dagstuhl seminar "Formal Protocol Verification Applied"*, page 117, 2007.

[26] Mike Bond and Ross Anderson. API-level attacks on embedded systems. *IEEE Computer*, 34(10):67–75, 2001.

[27] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei Skorobogatov, and Ross Anderson. Chip and Skim: cloning EMV cards with the pre-play attack. *IEEE Symposium on Security and Privacy (SP)*, 2014.

[28] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. SoK: Research perspectives and challenges for Bitcoin and cryptocurrencies. *IEEE Symposium on Security and Privacy (SP)*, 2015.

[29] Michael Burrows, Martin Abadi, and Roger M. Needham. A logic of authentication. *Proceedings of Royal Society*, pages 233–271, 1989.

[30] Vitalik Buterin. Selfish mining: A 25% attack against the Bitcoin network. 2013. URL `https://bitcoinmagazine.com/articles/selfish-mining-a-25-attack-against-the-bitcoin-network-1383578440/`.

[31] Miles Carlsten, Harry Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of Bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 154–167. ACM, 2016.

[32] Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[33] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[34] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In *Proceedings on Advances in cryptology*, pages 319–327. Springer-Verlag New York, Inc., 1990.

[35] Yao Chen, Radu Sion, and Bogdan Carbunar. XPay: Practical anonymous payments for Tor routing and other networked services. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 41–50. ACM, 2009.

[36] Daryl Collins, Jonathan Morduch, Stuart Rutherford, and Orlanda Ruthven. *Portfolios of the poor: how the world's poor live on $2 a day*. Princeton University Press, 2009.

[37] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[38] Wei Dai. B-money. 1998. URL `http://www.weidai.com/bmoney.txt`.

[39] Christian Decker and Roger Wattenhofer. Information propagation in the Bitcoin network. In *IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, 2013.

[40] Jamie DeCoster. Scale construction notes. *Department of Psychology University of Alabama*, 2000. URL `https://www.researchgate.net/profile/Jamie_Decoster2/publication/264874105_Scale_Construction_Notes/links/540f0f740cf2f2b29a3dccbb.pdf`.

[41] Roger Dingledine and Paul Syverson. Reliable MIX cascade networks through reputation. In *International Conference on Financial Cryptography and Data Security*, pages 253–268. Springer, 2002.

[42] Roger Dingledine and Dan S. Wallach. Building incentives into Tor. In *International Conference on Financial Cryptography and Data Security*, pages 238–256. Springer, 2010.

[43] Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. A reputation system to increase mix-net reliability. In *International Workshop on Information Hiding*, pages 126–141. Springer, 2001.

[44] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies*, pages 67–95. Springer, 2001.

[45] Roger Dingledine, Nick Mathewson, and Paul Syverson. Reputation in P2P anonymity systems. In *Workshop on economics of peer-to-peer systems*, volume 92, 2003.

[46] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[47] Ronald G. Downey and Craig V. King. Missing data in Likert ratings: A comparison of replacement methods. *The Journal of general psychology*, 125(2):175–191, 1998.

[48] Saar Drimer and Steven J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, pages 7:1–7:16, Berkeley, CA, USA, 2007. USENIX Association. ISBN 111-333-5555-77-9. URL `http://dl.acm.org/citation.cfm?id=1362903.1362910`.

[49] Pascaline Dupas, Sarah Green, Anthony Keats, and Jonathan Robinson. Challenges in banking the rural poor: Evidence from Kenya's western province. Technical report, National Bureau of Economic Research, 2012.

[50] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology*, pages 139–147. Springer, 1993.

[51] Martin Emms, Budi Arief, Leo Freitas, Joseph Hannon, and Aad van Moorsel. Harvesting high value foreign currency transactions from EMV contactless credit cards without the PIN. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 716–726. ACM, 2014.

[52] EMVCo. *EMV – Integrated Circuit Card Specification for Payment Systems, Book 1: Application Independent ICC to Terminal Interface Requirements*, 4.3 edition, 2011.

[53] EMVCo. *EMV – Integrated Circuit Card Specification for Payment Systems, Book 2: Security and Key Management*, 4.3 edition, 2011.

[54] EMVCo. *EMV – Integrated Circuit Card Specification for Payment Systems, Book 3: Application Specification*, 4.3 edition, 2011.

[55] EMVCo. *EMV – Integrated Circuit Card Specification for Payment Systems, Book 4: Cardholder, Attendant, and Acquirer Interface Requirements*, 4.3 edition, 2011.

[56] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

[57] Laura Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3): 379–383, 2003.

[58] Mainak Ghosh, Miles Richardson, Bryan Ford, and Rob Jansen. A TorPath to TorCoin: proof-of-bandwidth altcoins for compensating relays. In *Privacy Enhancing Technologies Symposium*, 2014.

[59] Steve Glassman. The Millicent protocol for inexpensive electronic commerce. 1995. URL `http://www.research.digital.com/SRC/millicent`.

[60] Steven Goldfeder, Harry Kalodner, Dillon Reisman, and Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2018.

[61] Laura Gray. Are you ready for 30 June 2018? saying goodbye to SSL/early TLS. 2017. URL `https://blog.pcisecuritystandards.org/are-you-ready-for-30-june-2018-sayin-goodbye-to-ssl-early-tls`.

[62] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144, Washington, D.C., 2015. USENIX Association. ISBN 978-1-931971-232. URL `https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman`.

[63] Danny Yuxing Huang. *Using Crypto-currencies to Measure Financial Activities and Uncover Potential Identities of Actors Involved*. PhD thesis, UC San Diego, 2017. URL `https://escholarship.org/uc/item/6jk6w02v`.

[64] Nick Hughes and Susie Lonie. M-PESA: mobile money for the "unbanked" turning cellphones into 24-hour tellers in Kenya. *Innovations: Technology, Governance, Globalization*, 2(1-2):63–81, 2007.

[65] GSMA Intelligence. The Mobile Economy–Africa 2016. *London: GSMA*, 2016. URL `https://www.gsma.com/mobileeconomy/africa/`.

[66] William Jack and Tavneet Suri. Mobile money: The economics of M-PESA. Technical report, National Bureau of Economic Research, 2011.

[67] Rob Jansen, Nicholas Hopper, and Yongdae Kim. Recruiting new Tor relays with BRAIDS. In *Proceedings of the 17th ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 319–328. ACM, 2010.

[68] Rob Jansen, Nicholas Hopper, and Yongdae Kim. LIRA: Lightweight incentivized routing for anonymity. In *Network and Distributed System Security Symposium (NDSS)*, 2013.

[69] Rob Jansen, Andrew Miller, Paul Syverson, and Bryan Ford. From onions to shallots: Rewarding Tor relays with TEARS. In *Privacy Enhancing Technologies Symposium*, 2014.

[70] Stanisław Jarecki and Andrew Odlyzko. An efficient micropayment system based on probabilistic polling. In *International Conference on Financial Cryptography and Data Security*, pages 173–191. Springer, 1997.

[71] Ronald Kainda, Ivan Flechais, and A.W. Roscoe. Usability and security of out-of-band channels in secure device pairing protocols. In *Fifth Symposium on Usable Privacy and Security (SOUPS 2009)*, page 11. ACM, 2009.

[72] Ghassan O. Karame, Elli Androulaki, and Srdjan Capkun. Double-spending fast payments in Bitcoin. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 906–917. ACM, 2012.

[73] Jake Kendall, Johanna Godoy, Robert Tortora, and Jan Sonnenschein. Payments and money transfer behavior of Sub-Saharan Africans. 2012.

[74] Joshua A. Kroll, Ian C. Davey, and Edward W. Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, 2013.

[75] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems (TOCS)*, pages 133–169, 1998.

[76] Ben Laurie. Decentralised currencies are probably impossible–but let's at least make them efficient. 2011. URL `https://www.links.org/files/decentralised-currencies.pdf`.

[77] Ben Laurie. An efficient distributed currency. 2011. URL `https://www.links.org/files/distributed-currency.pdf`.

[78] Raph Levien. Attack-resistant trust metrics. In *Computing with Social Trust*, pages 121–132. Springer, 2009.

[79] James R. Lewis and Jeff Sauro. The factor structure of the system usability scale. In *International conference on human centered design*, pages 94–103. Springer, 2009.

[80] Ritch Macefield. How to specify the participant group size for usability studies: a practitioner's guide. *Journal of Usability Studies*, 5(1):34–45, 2009.

[81] G. Maxwell. Merkle tree of open transactions for lite mode?, 2011. URL `https://bitcointalk.org/index.php?topic=21995.0`.

[82] Timothy May. The crypto anarchist manifesto. *Effector On line*, 1992.

[83] Timothy May. The cyphernomicon: cypherpunks FAQ and more. *Version 0.666, September*, 1994.

[84] Timothy May. Crypto anarchy and virtual communities. *Crypto Anarchy, Cyberstates, and Pirate Utopias*, pages 65–79, 2001.

[85] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of Bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 Conference on Internet Measurement Conference (IMC)*, pages 127–140. ACM, 2013.

[86] Cynthia Merritt. Mobile money transfer services: the next phase in the evolution of person-to-person payments. *Journal of Payments Strategy & Systems*, 5(2):143–160, 2011.

[87] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from Bitcoin. In *IEEE Symposium on Security and Privacy (SP)*, pages 397–411. IEEE, 2013.

[88] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocol–version 2. 2003. URL `https://gnunet.org/mixmaster-spec`.

[89] Tyler Moore and Nicolas Christin. Beware the middleman: Empirical analysis of Bitcoin-exchange risk. In *International Conference on Financial Cryptography and Data Security*, pages 25–33. Springer, 2013.

[90] Tim Moreton and Andrew Twigg. Trading in trust, tokens, and stamps. In *Proceedings of the First Workshop on Economics of Peer-to-peer Systems*, 2003.

[91] Malte Möser and Rainer Böhme. Trends, tips, tolls: A longitudinal study of Bitcoin transaction fees. In *Second Workshop on Bitcoin Research, affiliated with the International Conference on Financial Cryptography and Data Security, Puerto Rico*, 2015.

[92] Malte Möser, Rainer Böhme, and Dominic Breuker. Towards risk scoring of Bitcoin transactions. In *International Conference on Financial Cryptography and Data Security*, pages 16–32. Springer, 2014.

[93] Steven J. Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. Chip and PIN is broken. In *IEEE Symposium on Security and Privacy (SP)*, pages 433–446. IEEE, 2010.

[94] Steven J. Murdoch, Mike Bond, and Ross Anderson. How certification systems fail: Lessons from the ware report. *IEEE Security & Privacy*, 10(6):40–44, 2012.

[95] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[96] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[97] Jakob Nielsen and Thomas K. Landauer. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 206–213. ACM, 1993.

[98] Julius Okello. Effectiveness and challenges of using mobile money service in the implementation of the social assistance grants for empowerment programme in Uganda. *IMTFI Final Report*, 2015.

[99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[100] Jean Ray. *Malpertuis*, volume 142. Marabout, 1962.

[101] Fergal Reid and Martin Harrigan. *An analysis of anonymity in the Bitcoin system*. Springer, 2013.

[102] Ronald L. Rivest. Electronic lottery tickets as micropayments. In *International Conference on Financial Cryptography and Data Security*, pages 307–314. Springer, 1997.

[103] Ronald L. Rivest and Butler Lampson. SDSI–a simple distributed security infrastructure. Crypto, 1996.

[104] Ronald L. Rivest and Adi Shamir. PayWord and MicroMint: Two simple micropayment schemes. In *International workshop on security protocols*, pages 69–87. Springer, 1996.

[105] Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[106] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in Bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 515–532. Springer, 2016.

[107] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from Bitcoin. In *IEEE Symposium on Security and Privacy (SP)*, pages 459–474. IEEE, 2014.

[108] Jeff Sauro. *A practical guide to the system usability scale: Background, benchmarks & best practices*. Measuring Usability, 2011.

[109] Jeff Sauro. How to measure learnability. 2013. URL `http://www.measuringu.com/blog/measure-learnability.php`.

[110] Jeff Sauro and James R. Lewis. When designing usability questionnaires, does it hurt to be positive? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2215–2224. ACM, 2011.

[111] Jeff Sauro and James R. Lewis. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

[112] Clay Shirky. The case against micropayments. *OpenP2P O'Reilly*, 2000.

[113] Clay Shirky. Fame vs fortune: Micropayments and free content. *First published Sep*, 5:2003, 2003.

[114] Clay Shirky. Why small payments won't save publishers. 2009. URL `http://www.shirky.com/weblog/2009/02/why-small-payments-wont-save-publishers`.

[115] Manny Trillo. Stress test prepares VisaNet for the most wonderful time of the year, 2013. URL `http://www.visa.com/blogarchives/us/2013/10/10/stress-test-prepares-visanet-for-the-most-wonderful-time-of-the-year/index.html`.

[116] Jordi van den Breekel, Diego A. Ortiz-Yepes, Erik Poll, and Joeri de Ruiter. EMV in a nutshell. Technical report, 2016.

[117] Marie Vasek, Micah Thornton, and Tyler Moore. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In *International Conference on Financial Cryptography and Data Security*, pages 57–71. Springer, 2014.

[118] Robert A. Virzi. Refining the test phase of usability evaluation: How many subjects is enough? *Human factors*, 34(4):457–468, 1992.

[119] Visa. Visa service disruption. 2018. URL `https://www.visaeurope.com/newsroom/news/visa-service-disruption`.

[120] N. Walubengo. The mobile money apocalypse; what would happen to your money? 2012. URL `http://pesatalk.com/the-mobile-money-apocalypse-what-would-happen-to-your-money`.

[121] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas Hopper. rBridge: User reputation based Tor bridge distribution with privacy preservation. In *Network and Distributed System Security Symposium (NDSS)*, 2013.

[122] David Wheeler. Transactions using bets. In *International Workshop on Security Protocols*, pages 89–92. Springer, 1997.

[123] Paul Youn, Ben Adida, Mike Bond, Jolyon Clulow, Jonathan Herzog, Amerson Lin, Ronald L. Rivest, and Ross Anderson. Robbing the bank with a theorem prover. Technical report, University of Cambridge, Computer Laboratory, 2005. URL `https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-644.pdf`.

[124] Jamie M. Zimmerman and Silvia Baur. Understanding how consumer risks in digital social payments can erode their financial inclusion potential. Technical report, World Bank, Washington DC, 2016. URL `https://openknowledge.worldbank.org/bitstream/handle/10986/24568/Understanding000inclusion0potential.pdf`.