Alphacodes: Usable, Secure Transactions with Untrusted Providers using Human Computable Puzzles

Ashlesh Sharma[1], Varun Chandrasekaran[2], Fareeha Amjad[3], Dennis Shasha[2] and Lakshminarayanan Subramanian[2]

[1]ashlesh@entrupy.com; Entrupy Inc.

[2]vc1113@nyu.edu, {shasha,lakshmi}@cs.nyu.edu; New York University

[3]fareeha.amjad@nyu.edu; New York University, Abu Dhabi

ABSTRACT

Many banking and commerce payment systems in developing regions require users to share private or sensitive information in clear-text with untrusted providers, exposing them to different forms of man-in-the-middle attacks. In this paper, we introduce *Alphacodes*, a new paradigm that enables users to secure transactions with untrusted parties using the notion of *human-computable visual puzzles.* We describe how Alphacodes can be applied in different use cases and also explain two simple applications that we have built using this framework. We motivate our solution using security vulnerabilities in existing systems, and show how our protocol overcomes them. We demonstrate the ease of use of Alphacodes with minimal training using two simple crowdsourcing studies. Using another simple real world user study involving 10 users who speak Kannada (a regional Indian language), we show that the Alphacodes paradigm can be easily extended to languages beyond English.

CCS Concepts

•Security and privacy → Usability in security and privacy; •Human-centered computing → *Interaction techniques;*

Keywords

Usable Security, Human Computation Puzzles, Branchless Banking, Alphacodes

1. INTRODUCTION

Most forms of financial transactions including remittance, money transfers, branchless banking, and phone based transactions in developing regions are susceptible to a wide range of security threats - from untrusted providers and fraudulent middle-men.

ACM DEV '16, November 17 - 22, 2016, Nairobi, Kenya

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4649-8/16/11...$15.00

DOI: http://dx.doi.org/10.1145/3001913.3001924

Most of these transactions are performed over *clear-text channels without the use of cryptographic operations of any sort.* The user may expose highly sensitive information (such as identity information, credit card numbers, bank accounts etc.) in clear-text to an untrusted provider or a middle-man to complete the transaction, thereby exposing the user to different attacks including data corruption, identity theft, replay attacks, and identity spoofing, affecting the integrity and authenticity of the transaction. To elaborate further, consider the following forms of real-world transactions:

1. *Phone-based transactions*: Phone-based transactions where users share credit-card numbers or bank accounts over a phone line for convenience.
2. *Remittances and Money Transfers:* Remittance transactions where users interact with untrusted agents to send or receive money [10].
3. *Branchless banking:* Users in rural areas interact with untrusted shopkeepers (who act as agents of banks) to perform banking transactions to deposit or withdraw money from their bank accounts.

While there have been significant advances in cryptographic protocols for financial transactions, our setting differs from conventional transaction modalities in one key aspect: *the human user performing the transaction is dependent on another human who acts as the provider (such as an agent or operator or shopkeeper) to enable the financial transaction is required.* In essence, the human performing the transaction is necessitated to reveal identity, account, and transaction credentials in clear-text to the untrusted provider. Such types of clear-text transactions are also prevalent in developed regions, where users performing e-commerce transactions on unknown, and potentially non-trustworthy Websites may reveal their credit-card and CVV/PIN information. The key question we aim to address in this paper is: *Can we develop a simple and usable security mechanism devoid of any cryptographic operations, where the human end-user can easily verify the authenticity of a transaction performed with an untrusted provider or agent?* The requirement of being *devoid of cryptographic operations* is a constraint imposed by the need to not change the status-quo of how people perform transactions today. Providing a dongle (or hardware token) to such a populus modifies the status-quo, and requires a large bootstrapping cost in teaching them how to utilize it. We require a security mechanism that is inherently *human-computable* (like passwords) but which can provide three key properties: (a) a strict notion of an end-to-end security guarantee for a transaction; (b) be incrementally deployable without changes to transaction modalities; (c) be very simple to use.

In this paper, we present the *Alphacodes* paradigm, a human-computable protocol devoid of any complex cryptography. Alphacodes are a set of simple code tables/books (or human-computable visual puzzles) shared between various interacting parties. When one party wants to conduct a transaction, the party performs the encoding of specific parameters using the simple task of pattern-matching. The transaction succeeds only if the counter-party, upon decoding the messages of the transaction, validates certain parameters. It is important to note that the set of Alphacodes[1] are a secret between the parties, and are for *one-time* use only. This ensures that MitM attacks fail. Each Alphacode is a spin-off of the popular Message Authentication Code (MAC) paradigm. Alphacodes are analogous to short verification codes (replacement for randomly generated confirmation codes - which don't really solve for the issues stemming from the human-human clear-text channel). Unlike traditional authentication mechanisms which primarily just authenticate identities, Alphacodes can be used for authenticating the important transaction digits in each transaction.

We note that any security protocol involving human computation would require out-of-band communication of secret information between the user and a trusted third party (to authenticate the transaction). If every message is signed by participating parties using cryptographic keys, the underlying problem in many of these cases can be easily solved. However, in many of the settings outlined above, the clear-text human-human communication channel is devoid of cryptography, and the goal of the user is to generate a *simple, human-computable proof* that guarantees the authenticity of a transaction to the user.

Contributions: We demonstrate (and prove) the security of Alphacodes against MitM attacks, and replay attacks. The simplicity of the attacks described coupled with the ease of launching them drove us to rethink the existing security primitives in these regions. The ease of generation and solvability of Alphacodes makes it a viable solution, and favorable for adoption in existing applications. Based on experiments conducted on Amazon Mechanical Turk, we evaluate the usability of the Alphacodes paradigm based on ease of solvability, and time taken for the same. We observe that on average, a user could quickly create the encoding in under 20 seconds. We also observe that a majority of the users solved every puzzle accurately. We also observe that with minimal training, users are able to adapt the Alphacodes paradigm to a language different from English (Kannada). Alphacodes can be used as a building block for applications and protocols that can be used in a variety of settings, from branchless banking application in rural areas to e-commerce transactions over the Web.

2. MOTIVATION

As mentioned earlier, revealing sensitive transaction and identity information in human-provider transaction ecosystems is fairly common practice. While readers may be fairly familiar with the problems associated with fraudulent websites and phone-based credit card transactions, we provide some broader context to other lesser known clear-text transaction modalities which are common in developing regions.

1. A Case For Branchless Banking: A study by CGAP and the World Microfinance Forum on financial inclusion states that the leading state-owned banks including the Postal Savings Bank of China and the Agricultural bank of China boast enormous numbers of retail bank accounts, at 475 million and 320 million respectively. This translates to 33% of all Chinese citizens holding an account at one of these two banks. The utility of these bank accounts, however, is the real question. There is a clear recognition that the poor, particularly those in remote areas, have trouble accessing accounts, and use them mainly for encashment. However, this often requires costly travel to bank branches in distant cities and towns. Thus, the poor have to ultimately turn to local merchants for loans, often at exorbitant rates. On the other hand, due to lack of financial incentive, the traditional brick-and-mortar banks do not have branches in rural areas. To mitigate these issues, branchless banking initiatives are being implemented in parts of the developing world.

Recently, finance and security experts have raised concerns about the lack of security in rural banking models [8, 19]. People are not certain whether it is a safe way to conduct financial transactions [10]. Higher security might bring more users to use rural banking, but complex protocols may frustrate or confuse customers. Therefore, it is important to provide secure but simple to use protocols to conduct banking transactions in remote areas.

2. Money Transfers & Remittances: A significant fraction of families in developing regions rely on remittances from household members elsewhere for subsistence. Most money transfer mechanisms have relied on agent models to deliver money in rural contexts. In many of these money transfer mechanisms, the final step involves the user revealing most of the sensitive information to the agent for completing the transaction. While SMS-based mobile banking has enjoyed wide acceptance in developing nations such as Kenya, Tanzania, Philippines, and India, the ground reality is that the user is still dependent on the agent to procure the money for a variety of reasons including semi-literacy, and lack of technological proficiency. For example, M-Pesa in Kenya has adopted over 85,000 qualified agents throughout the country who form the backbone of the mobile banking industry. However, the security in SMS based mobile banking is minimal with several known instances of security breaches [19, 20]. Also note that in the case of mobile banking, encryption does not solve the last hop human-human interaction problem where the human provider may not be trustworthy.

There has been very limited work in trying to secure branchless banking and money transfer applications. The work by Sharma et al. [24] uses simple nonces among the user, agent, and bank to provide secure protocols for deposit and withdrawal in a rural setting, where a farmer, through a shopkeeper, accesses a bank. In that scenario, the farmer does not trust the shopkeeper. Eko, a mobile banking service in India, uses a number matching scheme to authenticate users on a per transaction basis based on codebooks, as the user interacts with the bank [22]. Unfortunately, as the authors note, their authentication scheme does not guarantee

[1]Used analogously with code-tables, and the actual code produced, depending on context

strong security against adversaries who can spoof caller IDs or mount MitM attacks. The concept of codebooks has partially been used by Paik et al [19] to secure mobile banking transactions. However these are meant to defend again attacks on SIM authentication. We discuss additional related work in greater detail later in the paper (§8).

3. ALPHACODES

In this section, we present the basic idea underlying our solution to the problems discussed in §1. To begin summarizing, consider the context of banking for semi-literate populations. Through our protocol, we wish to authenticate transactions performed by the untrusted middle man using (one-time usage) codebooks which are populated with random entries. Using specific portions of the transaction (such as the amount), customers compute an authentication code using the codebook. Thus, our paradigm is analogous to a combination of the popular OTP and MAC paradigms.

We make the following assumptions, that govern the security of our protocol, namely:

- Humans performing the encoding using the codebook behave in an honest and rational way. They do not share their codebooks with anyone.
- The codebooks are populated with random entries such that the encoding of a n-digit number using two distinct codebooks produces the same output with negligible probability (i.e. collision resistance).
- An adversary can only compromise the channel used for transacting (by attempting MitM attacks), not the secure channel used for distribution or out-of-band correspondence.

As a simple example, consider Alice and Bob who wish to transact in a secure manner. They pre-share a pair of codebooks using some secure channel. By encoding a number using the codebook, Alice can send Bob the information assuring authenticity and integrity. To elaborate, both Alice and Bob have Tables 1 and 2 in their possession. These tables are kept secret between Alice and Bob. For the sake of simplicity, let us assume a peer-to-peer communication scenario where Alice wants to send the number 250 to Bob.

$$\text{Table 1} = \begin{bmatrix} & \text{I} & \text{II} & \text{III} \\ 0 & \text{YK} & \text{NP} & \text{CK} \\ 1 & \text{CT} & \text{RK} & \text{NC} \\ 2 & \text{HY} & \text{XA} & \text{XK} \\ 3 & \text{AN} & \text{MF} & \text{YB} \\ 4 & \text{RB} & \text{PL} & \text{VL} \\ 5 & \text{OE} & \text{CA} & \text{GS} \\ 6 & \text{JM} & \text{ER} & \text{ET} \\ 7 & \text{SZ} & \text{FH} & \text{MA} \\ 8 & \text{RI} & \text{WZ} & \text{BU} \\ 9 & \text{IP} & \text{UL} & \text{DO} \end{bmatrix} \text{Table 2} = \begin{bmatrix} & \text{I} & \text{II} & \text{III} \\ 0 & \text{DQ} & \text{MC} & \text{RF} \\ 1 & \text{VN} & \text{FC} & \text{EY} \\ 2 & \text{KJ} & \text{TQ} & \text{ML} \\ 3 & \text{AM} & \text{CH} & \text{BV} \\ 4 & \text{EC} & \text{LD} & \text{XM} \\ 5 & \text{CI} & \text{GT} & \text{ZG} \\ 6 & \text{XE} & \text{DF} & \text{PA} \\ 7 & \text{WI} & \text{ZV} & \text{OI} \\ 8 & \text{ZR} & \text{LQ} & \text{DQ} \\ 9 & \text{LF} & \text{ME} & \text{IS} \end{bmatrix}$$

Step 1: From left to right, Alice matches the i^{th} digit (say j) with the entry in i^{th} column, and in the row labeled j. To elaborate, 2 corresponds to codeword HY in row 2, column I; 5 corresponds to CA in row 5, column II; 0 corresponds to CK in row 0, column III. Thus, 250 is encoded as *HYCACK*.

Step 2: Alice sends both 250 and *HYCACK* to Bob. Bob decodes *HYCACK* to 250 by matching the letters to numbers using the Table 1 (shared earlier with Alice). Bob now has the information sent to him by Alice and can be confident that the possessor of Table 1 sent the message, because most sequences of six letters would fail to represent three numbers. For any $n-$digit number which is represented by $k-$character codeword per digit, the probability of generating another valid sequence (i.e. collision) is 26^{-kn} (for the English language).

Step 3: Similarly, Bob encodes 250 using Table 2 obtaining *KJGTRF*. He sends the encoding (and the number 250) to Alice across the same channel. Alice receives *KJGTRF* and decodes it to obtain 250 using Table 2. Alice is convinced that Bob obtained the correct message because of this challenge-response mechanism. As explained earlier, the probability of an adversary intercepting an encoding, and responding with the correct response is negligible.

Salient Features: The term encoding refers to the process of generating the appropriate verification code based on an input. Similarly, decoding refers to the reverse process. Each codebook comprises of multiple tables (as above), and *each table is used for one transaction only, ensuring uniqueness (and inability to replay).* To perform another transaction, another set of secret tables have to be shared (or agreed) between Alice and Bob. The protocol that is employed by Alice and Bob is essentially a one-time substitution cipher system. The table may have more columns to encrypt larger messages. Thus, Alphacodes have the following properties:

- *Authentication & Integrity*: Alphacodes support both message authentication and integrity. Only the owners of the codebooks can generate valid sequences for the information to be shared ensuring authentic information. Also, any accidental or deliberate tampering of information is detectable, enforcing message integrity.
- *Secure, Offline Distribution*: Alphacodes can be distributed in an offline fashion. For example, the bank could provide paper copies of Alphacodes, either in person or via secure courier. The customer can then login to the bank's site using the Alphacodes which would be known only to the bank and the user. Note that the secure courier (or other secure channels) can not be used for transaction purposes as they are (i) expensive to repeatedly use, and (ii) offline, and hence not interactive.
- *Low Cost*: Alphacodes are of lower cost than compared to other technologies such as two-factor authentication and hardware tokens, in terms of production and deployment [1].
- *Ease Of Use*: Alphacodes are easy to use because table lookup is a simple task. The Eko algorithm entails putting in one-time numbers interspersed with numbers from a personal identification number in various places. That is at least as difficult as the process described here, and their user tests indicate that even illiterate users can do this.

4. CROWDSOURCED EVALUATION

We conducted two experiments on Amazon's crowdsourcing platform, Mechanical Turk, to determine the ease of interpretation of Alphacodes. In addition, we conducted a small scale user study to test interpretability of Alphacodes in Kannada, a local Indian language. We note upfront that all these simple studies are designed as a *first order test* to see whether Alphacodes are indeed easy to use, with a small, concise set of instructions provided to guide participants. In the Amazon MTurk experiment, users were provided two solved questions at the beginning of each task as examples. In the local language experiment, there was minimal training of users with a small number of Alphacode examples in the local language. Next, we describe these two studies in detail and our corresponding observations from these studies.

4.1 MTurk Task Definition

As shown in the examples prior, each Alphacode table (or grid) was composed of 10 rows and n columns, where a n digit number was provided as input. We use $n = 4$ across most experiments, to ensure sufficient complexity in the tasks provided without overburdening the workers. In addition, the choices we provided as answers were meant to deliberately confuse the users, and were set as slight variants of the correct answer. We expect the workers to perform simple tasks including adhering to the instructions provided, perform pattern matching, and select the radio button corresponding to the correct answer. We believe that most workers act honestly due to strict regulations enforced by Amazon on deviants. In subsequent subsections, we describe the experiments in detail, and explain the results obtained.

4.2 MTurk Worker Selection

The experiments that we will describe in subsequent sections were open to MTurk workers around the world. For privacy reasons, Amazon does not disclose specific information regarding the workers' demographics, nor does it release their educational background and qualifications. Our study specifically did not collect information about worker demographics, educational background or qualifications.

4.3 MTurk: Solvability of Alphacodes

This experiment was specifically designed to test and observe the ease of adoption (via solvability) of the Alphacodes paradigm. To do so, our MTurk worker pool contained 27 participants. A collection of 540 unique puzzles were evenly distributed among them i.e. each user was given 20 puzzles to solve. 10 of these puzzles had English entries in the grid, and 10 had Non-English (or symbolic) entries. As explained earlier, each participant was provided the same set of concise instructions and two solved examples to enable a quick bootstrap for the paradigm. Subsequently, each participant was allotted 15 minutes to solve all 20 puzzles. We believed this time limit to be reasonable as solving each puzzle only entailed selecting the button corresponding to its solution after performing rudimentary pattern matching.

We found that on an average, each user took 6.8 minutes (with 8.2 minutes to spare) to complete the entire task which roughly translates to 20 seconds for solving each puzzle. The median and mode accuracy was 100% which means that the majority of the users got all puzzles correct and errors were committed by a small minority. Out of the 540 puzzles posed, the number of errors committed was higher in English-based puzzles (22 out of 270) when compared to symbol-based puzzles (15 out of 270). Despite strict guidelines imposed, we suspect that this was due to the lack of user interest in performing the repetitive task, or because of the remuneration to effort ratio [11]. We believe that the time allotted was not a factor, as the erroneous answers were provided well within the 15 minute limit. The average time taken to solve the puzzles indicates the simplicity of the paradigm proposed, suggesting easy adoption. As the users were not provided the same puzzle, we are unable to comment on the difficulty of individual puzzles. Surprisingly, the participants performed better on symbol-based puzzles. Even among those users who committed errors, we observed that as the participants observed more puzzles, the accuracy became slightly better. Since the non-English puzzles came after the English puzzles, we believe the solvability improved. Overall, we conclude that the participants understood how to use Alphacodes with limited training.

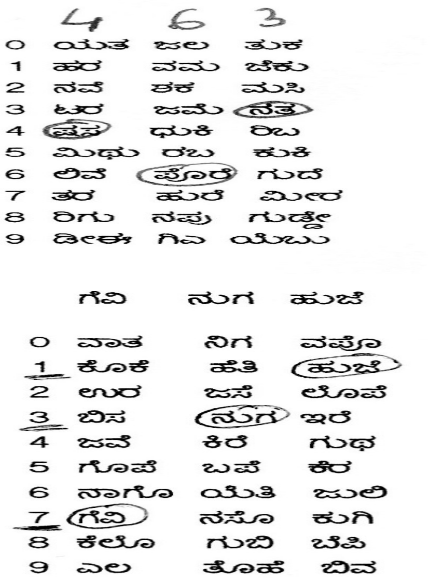
4.4 MTurk: Transactions Using Alphacodes

This experiment focused on understanding if users are capable of performing secure transactions using Alphacodes; this involved encoding the transaction amount into a character string. Though not the exact protocol, this test was performed with the intention of understanding if users are able to detect fraud during the transaction process. To elaborate, each user was given the task of performing a transaction of a specified amount. The amount was specified with two decimal digit accuracy and the total number of digits varied between 3 to 6. The users are expected to choose the $K = 4$ significant digits to solve the Alphacode puzzles. If the number of digits including the decimal places is less than 4, the user is expected to prepend the corresponding number of 0s after considering the decimal places. The user is provided multiple examples on how to choose the 4 significant digits. Each user is expected to perform 20 transactions, where each transaction involves solving two puzzles: the *Sender Puzzle* and the *Receiver Puzzle*. For the Sender puzzle, the participant is required to type the correct solution. In the Receiver puzzle, the user is specified a confirmation code as the solution and the participant is required to verify if the solution provided is correct or not. For 50% of randomly chosen transactions, we provided incorrect answers for the receiver puzzles which were very similar to the correct answer. For this experiment, our user pool had 18 members (not overlapping with the previous MTurk experiment). A collection of 20 transactions containing 40 unique puzzles (20 sender puzzles and 20 receiver puzzles) were distributed among them. Each user was given the same 40 puzzles to solve and was allotted 30 minutes.

We found that on the average, each user took 19.8 minutes to complete the entire task. We noted that the participants performed significantly better on receiver puzzles. The number of errors committed was higher in the sender puzzle category (28 out of 360) when compared to receiver puzzle category (6 out of 360). Unlike the first experiment, the sender puzzle error rate was marginally higher since we expected the users to type in the answer. However, similar to the previous case, we observed the median accuracy for both sender and receiver puzzles to be 100%. More than half the users got all 20 transactions completely correct, including both sender and receiver puzzles. Given that we observed an extremely high receiver puzzle solving accuracy despite giving incorrect codes for half the transactions, we strongly believe that the users did not have much difficulty understanding the Alphacode paradigm. We observe that the solvability of puzzles also got distinctly better as users solved more puzzles. Overall, we believe that despite minimal training of less than $2 - 3$ minutes, users could quickly and accurately perform transactions using Alphacodes. We admit that these are not full scale human-interface experiments but they provide encouraging evidence that the idea is feasible in practice.

4.5 Solvability In A Local Language

To demonstrate the ease of solvability of Alphacodes in a local language, we perform a different user study using the Kannada alphabet (Indian language in the state of Karnataka) with a small population of 10 users who all spoke the language. Note that Kannada is significantly more complex than English, with over 500 different written symbols in the alphabet.



	4	6	3
0	ಯುತ	ಜಲ	ತುಕ
1	ಹರ	ವಮ	ಜೆಕು
2	ನವೆ	ಶಕ	ಮುಸಿ
3	ಟರ	ಜಮು	[✓ ನೆತ]
4	[✓ ಷಸ]	ಧುಕಿ	ರಿಬ
5	ಮಿಥು	ರಬ	ಕುಕಿ
6	ಲಿವೆ	[✓ ಪೊರೆ]	ಗುದೆ
7	ತರ	ಹುರೆ	ಮೀರ
8	ರಿಗು	ನಪು	ಗುಡ್ಡೇ
9	ಡೀಈ	ಗಿಎ	ಯುಬು

	ಗೆವಿ	ನುಗ	ಹುಜೆ
0	ವಾತ	ನಿಗ	ವಪ್ರೊ
1	ಕೊಕೆ	ಹೆತಿ	[✓ ಹುಜೆ]
2	ಉರ	ಜಸೆ	ಲೊಪೆ
3	ಬಿಸ	[✓ ನುಗ]	ಇರೆ
4	ಜವೆ	ಕಿರೆ	ಗುಥ
5	ಗೊಪೆ	ಬಪೆ	ಕೆರ
6	ನಾಗೊ	ಯುತಿ	ಜುಲಿ
7	[✓ ಗೆವಿ]	ನಸೊ	ಕುಗಿ
8	ಕೆಲೊ	ಗುಬಿ	ಬೆಪಿ
9	ಎಲ	ತೊಹೆ	ಬಿವ

Figure 1: (a) One of the tables used as part of the user study. The participant successfully circles the letters based on the number 463, and (b) The participant successfully circles the letters based on the letters printed above the table and marks the corresponding row number.

Task 1: The participants were given a sheet of paper which consisted of an Alphacodes table in Kannada. At the top of the table, a three digit number was written and the users were told to match the three digit number with the corresponding letters as per the rules of the protocol. Each participant was individually presented the rules and a few sample tables were filled by us to show the participants the rules of the process. The teaching process took close to 5 minutes per participant, and they were told to fill up a new sheet based on the instructions provided, and the examples that they filled previously. Most (80%) of the participants were able to complete the task correctly within the specified time. We attribute the errors to the verbosity and (probable lack of) clarity in our explanations. The rest needed more time to understand the process, but they too completed the task after additional training.

Task 2: This task involved users matching the letters (written atop the table) with the corresponding numbers as per the pre-specified rules of the protocol. In this task, we increased the number of characters per digit to two Kannada characters. We first ask each user to pick a number, decipher the letters of the code and write the deciphered letters on top of the table. Once the letters are deciphered, the task is for each user to check whether the deciphered letters match the chosen number in the table. Similar to the first task, instructions were provided on an individual basis and sheets were filled up as part of the initial training process. Additional time was spent to make sure everyone understood the procedure. Once the training was complete, the users were given new sheet to fill up. All the users (100%) were able to satisfactorily complete the task as per the specified procedure. We believe that the training for the first task helped the users grasp the procedure better for the second task.

Figure 1(a) shows one of the tables used in the first study. The number 463 is written at the top of the table and the participant sifts through the first row (the number row) on the table to find 4, and circles the corresponding letters in the first column. Similarly, the participant circles the letters in the next two columns. Figure 1(b) shows an example table used in the second task, where the participant circles the letters that is printed on top of the table and marks the corresponding row number.

4.6 Takeaways & Limitations of Evaluation

Both these simple experiments demonstrate two powerful facts about Alphacodes: (a) The concept of Alphacodes can be easily extended to languages beyond English quite easily, and (b) The time taken for novice users to learn about the Alphacodes concept is minimal. Given repetitive practice, and the strong motivation behind using Alphacodes, users can quickly master the paradigm.

We would like to reiterate that these tests were designed to be a *first order study* of the usability of the Alphacodes paradigm. We believe that with improved instructions, a larger set of examples, and more practice, the percentage of failures can be reduced. We acknowledge the following limitations in our study, namely: (a) The test size (i.e. the number of participants) is not large enough across both tests, (b) The participants of our study at least in the MTurk studies may not comprise of the representative population, and (c) The study does not involve the encoding of a larger number of digits (than 4). We hope to address these limitations in future work.

5. ALPHACODES PROTOCOL

In this section, we formally introduce a three-party protocol which uses Alphacodes that enables a user to perform a human transaction with an untrusted (human) provider.

5.1 Operational Ecosystem & Threat Model

The basic setup is one where a user U wishes to transact through an untrusted provider P (such as a branchless banking provider, or banking agent), where P conducts the transaction with a trusted authority A (such as bank or credit card company). A issues identity credentials to U (and P in certain cases) used to perform these transactions after validating credentials of U (and P). The threat stems from the fact that U is reliant on P to perform the transaction and must share all the transaction details with P; *U does not directly communicate with A for the transaction.*

To complete a real world transaction, it is essential for P to directly contact A. In essence, U and P perform a clear-text human-human transaction, and P needs to communicate with A over any channel to complete the transaction. Hence P is a *middle-man* in the transaction.

In our threat model, U may expose critical information to P (in the clear) which may enable P to launch various types of attacks including identity theft, replay attacks etc. In addition, the communication channel between U and P, or between P and A could be susceptible to eavesdroppers.

We make the following assumptions with respect to the actors, and protocol:

1. U and A behave in an honest manner. Deviation from this behavior will produce unfavorable results for both.
2. The communication channel between A and P is secure and trustworthy in one direction.
3. The communication channel between U and A is secure and trustworthy in both directions.
4. All of the actors have unique identities issued to them. These identities are used as part of the transaction, either in clear or in encoded form.

5.2 The Three Party Protocol

The actors are denoted by the letters U for user, S for service provider, and A for the trusted authority. Based on assumptions discussed in §5, the trusted authority issues each service provider a numerical identifier (N_S). In our setup, U interacts with S, but wants A to perform the required authentication. This entails three distinct communication channels, one between each of the actors.

Additional Details: Observe that both U and S receive information from A (whether directly or through a third party) in a form that only A could send, maintaining message integrity. That information establishes approval (commit) or disapproval (abort) of the transaction. Once A approves, the transaction between U and S is completed.

Setup: Initially, both U and S register with A before performing any transaction. Upon registration, A separately issues a collection of Alphacodes to U and S. In the presence of such a centralized authority, the number of Alphacodes required to communicate is of the order $O(N + M)$, for N users and M service providers. The booklet given to U contains two sets of Alphacode tables, namely { $UT1_1$, $UT1_2$, ..., $UT1_n$ }, as well as { $UT2_1$, $UT2_2$, ..., $UT2_n$ }. Similarly, the booklet provided to S will contain tables{ $ST1_1$, $ST1_2$, ..., $ST1_m$ } and { $ST2_1$, $ST2_2$, ..., $ST2_m$ }. These booklets are good for n and m unique transactions respectively. The first set of tables $(UT1, ST1)$ are used for encoding transaction details whilst the second set of tables $(UT2, ST2)$ are used to check if the transaction has been committed or not. We will explain this in greater detail later in the section. We stress once again that each Alphacode table is used for only a single transaction.

Secure Channels: To share the code-tables with the users, the authority may resort to communication via secure avenues such as a courier. Alternatively, the user could pick up the code-tables from the authority from a secure location. This is imperative for the successful functioning of the paradigm, so as to ensure that there is no duplication of the code-tables, nor any misuse by any malicious party. This scenario is analogous to receiving check-books from banks, requiring no infrastructural change.

5.3 Protocol Steps

$$\begin{array}{ll}
U: & \text{Compute } \delta_i = \mathit{Alphacode}(Am, N_S, UT1_i) \\
U \rightarrow A: & \mathit{uname}, Am, \delta_i \\
U \rightarrow S: & \mathit{uname}, Am \\
S: & \text{Compute } \gamma_j = \mathit{Alphacode}(Am, ST1_j) \\
S \rightarrow A: & \mathit{uname}, Am, \delta_i, \mathit{sname}, \gamma_j
\end{array}$$

1. Initially, U provides S with enough information in the clear for S to understand the transaction amount Am, and metadata such as identities of interacting parties (collectively referred to as *uname*). Additional information could also be provided; this includes any form of specifics regarding the item being purchased. Similarly, S sends to A enough information to ensure that A can verify that U and S are asserting the same description of the transaction. We also note that if either U or S has a trusted channel (to economically and repeatedly use) to communicate with A, it can avoid using an Alphacode and send content directly to A. As explained in the steps above, U generates δ_i based on Am and the table $UT1_i$, by performing operation $\mathit{Alphacode}()$, which is described as follows: If the p^{th} position of Am is a, then the p^{th} position of δ_i is the string in $p + 1^{th}$ column and the $a + 1^{th}$ row of $UT1_i$. For example, let $Am = 21$, $N_S = 47$, and for simplicity,

$$UT_i = \begin{bmatrix}
0 & KL & WR & IU & AK & DO & HG & SP & GZ & WM \\
1 & HW & TU & BQ & GN & QL & JD & QB & LZ & UU \\
2 & YT & IP & NZ & LW & ND & EL & HD & JI & GX \\
3 & AD & HV & EI & YU & XO & QM & FJ & HR & GD \\
4 & WO & CK & DO & ER & AP & IL & PR & RH & HK \\
5 & RN & BA & LF & WV & EM & OX & KS & LT & KM \\
6 & JL & KQ & HB & PI & GI & PZ & CT & OY & TL \\
7 & GD & GP & UA & NQ & ZJ & NA & OP & MT & KY \\
8 & QE & SX & LX & CI & DP & FI & UE & YW & PO \\
9 & RY & RH & FP & SL & JA & OC & RV & LG & NV
\end{bmatrix}$$

then $\delta_i = YTTUDONQ$.

Observe that the input for the codebook is $Am||N_S$, where $||$ denotes the concatenation operation. Once U computes δ_i, U sends *uname*, Am and δ_i to S. Similarly, S computes γ_j using Am and $ST1_j$ by performing operation $\mathit{Alphacode}()$. It sends *uname*, Am, δ_i, *sname* (name of the service), and γ_j to A. The input (i.e. $Am||N_S$) could also be padded with leading zeros. For example, 21 could be thought of as 0000021 having the Alphacode output as *KLWRIUAKDOELQBRHKY*.

2. Note, as before, that both U and A must use table $UT1_i$ for only one transaction. A must have some additional state information (such as timestamps) to keep track of transactions it has already conducted with U. The amount of state information required grows linearly with the number of transactions authorized (or rejected) by the authority. We argue that

linear growth is quite acceptable in a use case of user-generated transactions, coupled with storage available at a low cost. If a second transaction uses the same table and a different amount, it violates the uniqueness property of the codebook and could indicate an attempt at fraud. A must also check that the encoded N_S corresponds to the *sname* sent from the sender who purports to be S. Without this additional check, U might think it is paying S but a malicious S' profits instead. Normally, N_S might require many digits but we use just two for the purpose of explanation.

3. Next, A computes δ'_i using Am' (it has received from U), N_S, and $UT1_i$. Similarly, A computes γ'_j using Am'' (it has received from S) and $ST1_j$.

   (a) If $\delta_i \neq \delta'_i$, or if $\gamma_j \neq \gamma'_j$, or if the server name does not correspond to N_S encoded in δ_i, then A computes $\alpha_i = Alphacode(00, UT2_i)$ and $\beta_j = Alphacode(00, ST2_j)$. This means that the first parameter of the operation $Alphacode()$ would be two zeros instead of Am. These two zeros signal A to abort the transaction. If Alphacodes are reused in transactions, A aborts them. The only exception to this occurs when A receives exactly the same message from S as it did in an earlier transaction (retransmission), in which case A will not commit or abort any transaction, but will send the response message it had sent earlier.

   (b) If $\delta_i = \delta'_i$, $\gamma_j = \gamma'_j$, and neither has been used for another transaction, A computes α_i using Am (which is proven to be the same as Am') and $UT2_i$ and computes β_j using Am (which is proven to be the same as Am'') and $ST2_j$. A sends α_i and β_j to S. S sends α_i to U. A also commits the transaction at this point even though U and S do not yet know the transaction is committed. Committing means that the trusted authority declares this transaction to be complete and, for example, might perform a money transfer.

$$
\begin{array}{ll}
A: & \text{Compute } \delta'_i = Alphacode(Am, N_S, UT1_i) \\
A: & \text{Compute } \gamma'_j = Alphacode(Am, ST1_j) \\
\text{if}: & \delta_i \neq \delta'_i \quad \text{or} \quad \gamma_j \neq \gamma'_j \\
 & A: \quad \text{Compute } \alpha_i = Alphacode(00, UT2_i) \\
 & A: \quad \text{Compute } \beta_j = Alphacode(00, ST2_j) \\
\text{if}: & \delta_i = \delta'_i \quad \text{and} \quad \gamma_j = \gamma'_j \\
 & A: \quad \text{Compute } \alpha_i = Alphacode(Am, UT2_i) \\
 & A: \quad \text{Compute } \beta_j = Alphacode(Am, ST2_j) \\
A \to S: & \alpha_i, \beta_j \\
S \to U: & \alpha_i
\end{array}
$$

4. S computes β'_j using Am and $ST2_j$. U computes α'_i using Am and $UT2_i$. If $\alpha_i \neq \alpha'_i$ for U or $\beta_j \neq \beta'_j$, then the transaction has been rejected by A. That is, if the encoded text does not correspond to any possible sequence of code-words, then the party receiving such a malformed message simply requests a retransmission. If the response is well-formed then it should either correspond to the amount originally sent or should be all zeros. If it is the latter case of all zeros, then the receiving party should note that the transaction has been aborted.

$$
\begin{array}{ll}
S: & \text{Compute } \beta'_j = Alphacode(Am, ST2_j) \\
U: & \text{Compute } \alpha'_i = Alphacode(Am, UT2_i) \\
\text{If (for U)}: & \alpha_i \neq \alpha'_i \\
 & \quad \text{Transaction has been aborted by A} \\
\text{If (for S)}: & \beta_j \neq \beta'_j \\
 & \quad \text{Transaction has been aborted by A} \\
\text{If (at U)}: & \alpha_i = \alpha'_i \\
 & \quad \text{U knows that transaction} \\
 & \quad \text{has been committed by A} \\
\text{If (at S)}: & \beta_j = \beta'_j \\
 & \quad \text{S knows that transaction} \\
 & \quad \text{has been committed by A}
\end{array}
$$

5.4 Protocol Over a Noisy Channel

In the face of message loss in the communication channel, good protocol design necessitates the ability to resend any lost message. A must keep track of transactions that have been executed and never execute the same transaction (in response to the same messages from U and/or S) more than once. As explained earlier, this can be implemented trivially by saving the transactions in a serial order within a database. The authority should store information regarding the relationships between messages received with the corresponding transactions performed, and always looking up messages received against those in the database. On the other hand, though A will not re-execute the transaction corresponding to a previously received message, it will repeat its response to other parties.

5.5 Analysis

In this subsection, we wish to highlight some of the salient features from the above discussion.

1. If $\delta_i \neq \delta'_i$ or if $\gamma_j \neq \gamma'_j$ or *sname* does not correspond to N_S as encoded in δ_i, then A assumes adversarial presence. In this case, A aborts the transaction. If A sent *ABORT* or any other message in clear-text, the adversary might be able to change it and continue the transaction. It is hard for an adversary to decode an encoded response because the adversary doesn't have access to the correct Alphacode table (see Theorem 1). Denial of transaction attacks are dealt with in §5.4.

2. If $\delta_i = \delta'_i$ and $\gamma_i = \gamma'_i$ and *sname* corresponds to N_S, then A can be confident the message has been sent by U and S respectively.

3. S computes β'_j using $ST2_j$ and U computes α'_i using $UT2_i$. S and U check the authenticity of β_j and α_i as follows: If $\alpha_i \neq \alpha'_i$ but α'_i consists of some (not all) entries from the Alphacode table, then U knows that the transaction has been aborted. That is, the abortion takes place with clear knowledge that A sent the message. If α'_i does not consist of entries from the codebook, then U requests the message again. If after several retries, U continues to see this inequality, then U waits more, or has to go out-of-band to find the status of the transaction. Similarly for S.

5.6 Comparison with Related Efforts

Eko is a large branchless banking institution which uses SMS receipts for ensuring transaction authenticity. Eko uses 10-digit signatures in SMS based transactions that are a combination of the first unused nonce in the codebook (distributed securely) and a 4-digit PIN used to replace the diamond-shaped blanks (in order) in the nonce. The authors of [22] utilize the fact that the PIN is used in the signature in order, thus allowing an easy PIN recovery attack. Once the PIN is obtained, the agents can mount a variety of attacks. To alleviate PIN recovery attacks, Panjwani et al. [22] propose an improvement in the Eko construction. However, these authors note that their protocol is susceptible to different forms of MitM attacks as acknowledged in their paper. The security guarantees of the Alphacodes protocol relies on pre-distribution of codebooks - where each code is used only for one transaction and is discarded after. This property enables Alphacodes to provide stronger security guarantees in the face of MitM attacks and other forms of security threats. We show that the ability of an attacker to reverse engineer the code for a single transaction is a very low probability event that is a function of the length of the code. The same transaction inputs with a different Alphacode produces different signatures making it resilient to replay attacks. These additional security guarantees come at the cost of requiring pre-distribution of Alphacodes in bulk from the bank to the end-user. A followup work by Panjwani [21] shows an alternative protocol to authenticate the receipt of a transaction using missed calls. This work also suffers from specific forms of MITM attack vulnerabilities, while the authors propose specific mechanisms to reduce the MITM attack risk. The receipt authentication protocol setting completely differs from the Alphacode setting where the transaction will not be complete without the human-computed authentication code. [21] also suggests that human-assisted authentication has a usability barrier due to cognitive overload. Alphacodes aims to significantly reduce this cognitive load by simplifying the code generation process to a simple visual lookup in a table without requiring the user to mathematically compute any function.

5.7 Protocol Limitations

We acknowledge the following limitations in the Alphacodes protocol, namely:

- The security of the Alphacodes paradigm depends on the safety of the pre-distribution channel. An adversary could compromise the protocol by mounting a DoS attack in this channel, but that is extreme.
- Alphacodes are best suited in settings where the number of digits that need to be authenticated in a transaction is limited (ideally less than 10). The Alphacodes code length increases with an increase in the number of digits encoded which may introduce both additional human cognition load and potentially increase the error rates in human computation.

6. APPLICATIONS AND IMPLEMENTATION

This section explains how the Alphacode protocols can be applied in many real-world scenarios. We have implemented simple applications for two of these settings which demonstrate how Alphacodes can be used in some of these contexts.

6.1 E-Commerce Transactions

For e-commerce, the purchaser plays the role of U in the protocols of the previous section, the vendor plays the role of S, and the bank or credit card provider plays the role of A. The registration and the steps of the protocols are the same as the protocols in §5. Transactions made by the purchaser are validated by the bank or credit card provider in the absence of any malicious activity from the vendor.

We implemented a simple setup which emulates this three party transaction scenario. Here, a trusted third party (A) issues identities to the user, distributes Alphacodes securely in bulk to the user (U), and verifies the transaction from the untrusted website (S). We specifically consider the case where a user aims to perform an e-commerce transaction on the Web using Alphacodes integrated with credit cards. Here, the Alphacode puzzles are similar to the use of per-transaction CVV codes or PIN numbers used in credit cards. We assume that A is responsible for issuing an unique identity to U, and there is secure channel for A to disseminate Alphacodes to U. We also assume that every Alphacode is associated with a unique puzzle number.

Consider U performing an e-commerce transaction with S. For simulation purposes, we created a simple e-commerce site (modeled after an online shopping site) where U can select several items in a shopping cart and perform an online transaction. When completing the transaction, U has to enter the user identity (issued by A), information about A, the puzzle number corresponding to a chosen Alphacode, and the solution to the sender puzzle based on the significant digits of the transaction amount. S needs to obtain authorization for the transaction amount from A, and hence needs to share the user information along with the Alphacode puzzle number and the sender code. A can accept or reject the transaction depending on whether the solution to the Alphacode is correct. If correct, A shares the recipient code response back to S, which then needs to be disseminated to U as the confirmation code for the transaction. If the confirmation code does not match the response from A, U can initiate a dispute resolution with A.

6.2 Mobile Banking

Another direct application is mobile banking where users can leverage their phones to perform banking transactions such as money transfer and payments. Consider the setup where one can use two independent channels for performing a transaction. A bank shares several Alphacodes (in bulk) with an end-user through a trusted channel. Using the Alphacode, the user follows the steps in the §5 to perform a transaction over a mobile channel such as SMS. Given that the individual codes are short, each of the 3 steps of the protocols can easily be encoded in an SMS message.

We implemented a simple example of using Alphacodes integrated with SMS-based banking. Here, we assume that all users (U) have a unique identity issued by a bank (A). The user identity is also directly linked to the mobile number of U. A runs a mobile money transfer service where users can send money from their account to another account using Alphacodes. In this case, we assume that Alphacodes are disseminated through an alternative secure channel where users can physically purchase Alphacodes similar to scratch cards (which are in essence pre-certified Alphacodes issued by A). The Alphacodes can be user-specific or user-agnostic. An Alphacode in isolation has no monetary value. Once a user is in possession of an unused Alphacode, the user sends an SMS for the transaction with the code. This message is sent to the mobile number corresponding to the bank. Our messages use the following simple format:

<amt>*<sID>*<rID>*<pID>*<**code**>

where *amt* is the amount, sID and rID are the identities of the sender and recipient, pID is the Alphacode puzzle number, and the **code** is the actual Alphacode used for the transaction. The bank checks the code and if it's correct sends the corresponding response code over SMS. If the response code does not match the expected response, the sender can initiate a dispute resolution with the bank.

6.3 Vocal Transactions & Phone Banking

We assume that phone lines constitute an insecure channel. The customer service representative plays the role of the service provider (S), the customer plays the role of the user (U), and the bank plays the role of the trusted authority (A). Given these roles, the protocols are the same as described in §5 except that S and U communicate over the phone while S will independently communicate with A (over any communication channel) to verify the transaction. S needs to forward the Alphacodes proof provided by A to U during the course of the phone-based transaction. We assume that A has pre-distributed Alphacodes to U through a trusted channel.

6.4 Rural Banking

Consider a scenario where a shopkeeper plays the role of the service provider (S), a farmer plays the role of the user (U), and the bank plays the role of the trusted authority (A). The protocols are the same as described in §5 except that there is a need to perform both deposits and withdrawals. A simple way to specify transaction nature involves representing deposits by 00 and withdrawals by 11. As part of the amount Am, the first two numbers (independent of the leading zeroes) would be either 00 or 11 depending on whether the farmer wants to deposit or withdraw money. The bank uses the encoding of the first two numbers in Am to determine the nature of the transaction. For example, if the farmer wants to deposit \$ 20, then Am would be 0020. Similarly if the farmer wants to withdraw \$ 20, then Am would be 1120.

7. SECURITY ANALYSIS

We begin our security analysis with a theorem that describes the low probability with which collisions occur using the Alphacodes paradigm. To elaborate:

Theorem 1. *For an Alphacode using the English alphabet, the probability of guessing an Alphacode used in a transaction correctly is $26^{-\ell}$, where ℓ is the size of the Alphacode for a given transaction input.*

Proof: Each entry in the code-table is a letter ranging from A to Z. Each entry is sampled uniformly at random, therefore the entries are independent of each other. Hence, the probability to correctly guess what an entry is, is $26^{-\ell}$. It is easy to observe that if the size of the alphabet used to create the tables increases, the probability of correctly guessing each entry decreases further. ■

To understand the larger set of attacks in the status quo, we consider GCASH, M-Pesa, mChek, Obopay, Zain, Cellpay, which are some of the well known players in the field of mobile banking. Work by Ignacio Mas of CGAP [3, 13] provides a breakdown of mobile banking based on the technologies currently in use (Voice, SMS, IVR etc) [13]. All the major mobile banking organizations either use SMS (Short Message Service), USSD (Unstructured Supplementary Service Data), STK (SIM ToolKit), Java based GPRS application, or IVR (Interactive Voice Response) and are fraught with issues [20]. We look at typical attacks such as replay, spoofing, phishing, man-in-the-middle, rootkit, and key stealing that plague and have the potential to disrupt mobile and branchless banking schemes and show how our scheme provides better, if not comparable security guarantees.

Replay Attacks: If the service provided is based on SMS, replay or outright forgery attacks are easy to facilitate [4, 27]. This is because most of the SMS in the developing world uses A5/0 or A5/2 algorithm for over-the-air encryption. In the case of A5/0, SMS is sent in clear-text to the base station from the cellphone. Even though the GSM voice traffic is encrypted (using A5/1 or A5/2), SMS is sent in clear-text [12]. It is important to note that though A5/2 has been phased out by the GSM Association, it is still in widespread use in developing regions. GCASH, Movibanco, and BAC Movil use SMS to send transaction details along with the secret PIN. A simple replay attack, where a message is intercepted and replayed multiple times to the service provider can be implemented with ease [2]. Alphacode protocols, on the other hand, thwart replay attacks as they can not be reused to perform a new transaction. The trusted authority detects reuse but does not execute the transaction again, as explained in §5.

Spoofing & Phishing: Though new SIM cards (Version 3) are harder to clone, older versions of SIM cards can be cloned [26]. Due to such cloning, mobile banking services dependent on SMS (such as GCASH) or SIM ToolKit (such as M-Pesa) break since the adversary has control of the secret key present in the SIM. It can pretend to be a client and conduct the transaction, regardless of the service provider. The Alphacode protocols are agnostic to the capabilities of the SIM. Even if a SIM is cloned, the adversary would need the code-tables, nonces, and unique identities which are distributed through a secure channel, to complete transactions.

8. RELATED WORK

Moran and Naor [14, 15], use physical envelopes in establishing a plausibly deniable polling scheme. They use basic cryptographic techniques and rigorously prove the security of their scheme under the UC (Universal Composable) model. The physical envelopes were a physical realization of an ideal definition of Distinguishing Envelopes (DE), which was described in their previous work on cryptographic protocols using *Tamper Evident Seals* [14]. This concept could be applied to alphacodes implemented as scratch cards. Once the surface is scratched off, anyone looking at the scratch card is aware that it has been scratched off. Moran and Naor [14, 15] use scratch cards for polling.

One-time Message Authentication Codes (MAC) provide an authentication scheme, where a one-time secret key is used to both compute and authenticate a message. The standard technique of computing one-time MACs uses pairwise-independent hash functions [25]. Alphacodes explains how to build a *humanly computable* one-time MAC where each scratch-card contains the secret key that can be used to authenticate a single transaction.

There has been some work in the area of using physical objects in cryptographic protocols. Fagin, Naor and Winkler [7] use physical objects such as cups, labels and a telephone system to share secret information between two parties. Naor, Naor and Reingold [18] propose a *zero knowledge proof of knowledge* protocol that uses newspaper and scissors to solve a kid's puzzle known as "Where's Waldo". Crepeau and Kilian [6] propose a protocol for sharing secret information between two parties using a deck of cards to play discreet solitary games.

Scheiner [23] used a deck of cards to implement *Solitaire cipher* that is used in communicating information between parties. Naor and Shamir [17] propose *Visual Cryptography* and implement Visual Authentication and Identification [16] protocols based on simple operations on binary images. They propose to implement their protocol using slides and transparencies. Chaum [5] proposes a new protocol for verifiable voting using secure paper ballots. These protocols (or variants) are hard to implement in the field due to two reasons. One, to use transparencies or slides in authenticating a transaction is impractical. Second, the protocols are too complex for users to comprehend for daily use.

Two-factor authentication is being used by a number of organizations to provide extra security to users [9]. Although it provides an additional layer of security, it does eliminate the problem of data leakage in unencrypted channels. The PIN acts as another password that has to be entered to gain access to the site, whereas secure data would still be vulnerable to a MitM attack.

9. CONCLUSIONS

Clear-text transactions between users and providers such as phone-based transactions, e-commerce in untrusted websites, remittances and branchless banking continue to remain common practice. This comes at the risk of exposing the users to different forms of MitM attacks. This paper aims to build new protocols based on the concept of Alphacodes, which are simple to use, and should prevent exploitation of the poor by unscrupulous merchants. The Alphacode protocols are designed to tolerate dropped messages, corrupt messages, MitM attacks, and other malicious attacks. and branchless banking contexts.

10. ACKNOWLEDGMENTS

We thank the NYU Abu Dhabi Research Institute and the Center for Technology and Economic Development (CTED) in NYU Abu Dhabi for supporting Ashlesh Sharma, Fareeha Amjad, and Lakshminarayanan Subramanian on this project. We would also like to thank Barath Raghavan, Kurtis Heimerl and the anonymous reviewers of DEV 2016 for their valuable feedback.

11. REFERENCES

[1] http://bit.ly/2cmFFBo.

[2] http://bit.ly/2cpn9sL.

[3] Ignacio Mas, Kabir Kumar. Banking on Mobiles: Why, How, for Whom? http://www.cgap.org/gm/document-1.9.4400/FN48.pdf.

[4] A. Biryukov, A. Shamir, and D. Wagner. Real Time Cryptanalysis of A5/1 on a PC. In *Proceedings of the 7th International Workshop on Fast Software Encryption*, FSE '00, pages 1–18, London, UK, 2001. Springer-Verlag.

[5] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2:38–47, January 2004.

[6] C. Crépeau and J. Kilian. Discreet solitary games. In *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 319–330, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[7] R. Fagin, M. Naor, and P. Winkler. Comparing information without leaking it. *Commun. ACM*, 39:77–85, May 1996.

[8] G. GCash. http://www.234next.com/csp/cms/sites/Next/Home/5413169-146/Branchless_banking:_Uncertainties_emerge.csp.

[9] E. Grosse and M. Upadhyay. Authentication at scale. *IEEE Security and Privacy*, 11:15–22, 2013.

[10] G. Ivatury and I. Mas. Early experiences with branchless banking. In *CGAP Focus Note 46*, Washington D.C., 2008.

[11] N. Kaufmann, T. Schulze, and D. Veit. More than fun and money. worker motivation in crowdsourcing-a study on mechanical turk. In *AMCIS*, volume 11, pages 1–11, 2011.

[12] S. Lord. Trouble at the Telco: When GSM goes bad. *Network Security*, January 2003.

[13] I. Mas. Economics of Branchless Banking. *Innovations*, Vol. 4, Issue 2, January 2009.

[14] T. Moran and M. Naor. Basing cryptographic protocols on tamper-evident seals. In L. C. et al., editor, *ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 285–297. Springer-Verlag, July 2005.

[15] T. Moran and M. Naor. Polling with physical envelopes: *a* rigorous analysis of a human-centric protocol. In S. Vaudenay, editor, *Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 88–108. Springer-Verlag, May 2006.

[16] M. Naor and B. Pinkas. Visual authentication and identification. In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 322–336, London, UK, 1997. Springer-Verlag.

[17] M. Naor and A. Shamir. Visual cryptography. pages 1–12. Springer-Verlag, 1995.

[18] M. Naor, N. Y, and O. Reingold. Applied kid cryptography: http://www.wisdom.weizmann.ac.il/ naor/papers/waldo.ps.

[19] M. Paik. Stragglers of the herd get eaten: security concerns for gsm mobile banking applications. In *Proceedings of the Eleventh Workshop on Mobile Computing Systems &#38; Applications*, HotMobile '10, pages 54–59, New York, NY, USA, 2010. ACM.

[20] S. Panjwani. Towards end-to-end security in branchless banking. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, pages 28–33. ACM, 2011.

[21] S. Panjwani. Practical receipt authentication for branchless banking. In *Proceedings of the 3rd ACM Symposium on Computing for Development*, page 3. ACM, 2013.

[22] S. Panjwani and E. Cutrell. Usable, low-cost, authentication for mobile banking. In *SOUPS*, 2010.

[23] B. Schneier. The solitaire encryption algorithm: http://www.schneier.com/solitaire.html.

[24] A. Sharma, L. Subramanian, and D. Shasha. Secure branchless banking. In *NSDR '09: Networked Systems for Developing Regions*, New York, NY, USA, 2009. ACM.

[25] D. R. Stinson. Universal hashing and authentication codes. In *International Crytology Conference*, pages 74–85.

[26] D. Wagner and I. Goldberg. . http://bit.ly/2ccvwYN.

[27] D. Wagner and I. Goldberg. The Real-Time Cryptanalysis of A5/2. *Rump Session Crypto '99*, 1999.