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

TR2016-982

Ashlesh Sharma, Entrupy Inc
Varun Chandrasekaran, New York University
Fareeha Amjad, New York University Abu Dhabi
Dennis Shasha, New York University
Lakshminarayanan Subramanian, New York University

Many banking and commerce payment systems, especially in developing regions, continue to 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 provides a usable security solution that enables users to perform secure transactions with untrusted parties using the notion of visual puzzles. Alphacodes are designed as verification codes for short message transactions and provide easy authentication of critical portions of a transaction. We describe how Alphacodes can be applied in different use cases and also show two simple applications that we have built using the Alphacodes framework. We show security vulnerabilities in existing systems and show how our protocol overcomes them. We also demonstrate the ease of use of Alphacodes with minimal training using two simple mechanical turk studies. Using another simple real world user study involving 10 users who speak Kannada (local Indian language), we show that the Alphacodes concept can be easily extended to other languages beyond English.

1. INTRODUCTION

Despite advances in security and cryptography, a non-trivial fraction of real-world commerce transactions continue to be conducted between users and providers in *clear-text*, where the user explicitly exposes highly sensitive identity information (such as credit card numbers, bank accounts etc.) in the clear to an untrusted provider. To complete the transaction, the provider requires this user information to authenticate the user's credentials with a trusted third party issuing the identity (such as banks, Visa/Master/Amex etc.). Revealing identity information to untrusted providers exposes the user to different types of attacks including data corruption, 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) *Untrusted websites:* E-commerce transactions on unknown, and potentially non-trustworthy websites where users may reveal their credit-card and CVV/PIN information;
(3) *Remittances and Money Transfer:* Remittance transactions where users interact with untrusted agents to send or receive money and often may get cheated on the amount of money being remitted by the recipient [11]. Remittances represents a very large market with significant fraction of transactions in certain under-developed countries and most remittance services rely on a network of human agents to correctly deliver cash to the end-user;
(4) *Branchless banking:* Branchless banking is a new form of banking in developing regions where banks work with shopkeepers to act as bank agents to provide financial access. In this case, users in rural areas interact with untrusted shopkeepers to perform banking transactions to deposit or withdraw money from their bank accounts.

In all these cases, the real threat is the user interaction with the provider under the assumption that the provider could be trustworthy; the user reveals all the relevant sensitive information to the provider, who can in turn potentially use this information for launching different forms of attacks on the user. Unfortunately, conventional security or cryptographic solutions may not be able to address this problem since the underlying channel could be encrypted or signed, but the real threat is the provider in the middle with the power to launch man-in-the-middle (MitM) attacks. To really address the provider-level threat, one needs a simple way to provide end-to-end security for each transaction to the user where the untrusted provider cannot perform any MitM attacks on the user. In other words, we require a *human authentication protocol* that enables each user to easily verify the end-to-end authenticity of any transaction in a simple manner, preferably without the need for any complex operations or constraints required on behalf of the user. Clearly, any such protocol would require an out-of-band offline communication of secret information between the user and the trusted third party. If every message is signed by each party using cryptographic keys, the underlying problem in many of these cases can be easily solved. However, in many of the settings outlined above, the goal of the user is to generate a *simple, human-computable proof* that guarantees the authenticity of a transaction between a user and an untrusted provider.

Threat Model: The basic setup that we aim to address comprises a user U, a provider P (such as e-commerce site or branchless banking provider or banking agent) and a trusted authority A (such as bank or credit card company) which issues identity credentials to U (and P in certain cases) to perform transactions between U and P. U and P directly interact over a channel to perform a transaction and P needs to interact with the authority A to complete an authorized transaction. We stress that to complete a real world transaction, it is essential for the commerce provider to directly contact the trusted authority (such as a bank). In our threat model, a user may expose critical information to a provider which may enable the provider to launch various types of attacks including identity theft, replay attacks etc. Ergo, U does not trust the provider P. In addition, we may also have the case where the communication channel between U and P or between P and A is susceptible to eavesdroppers from adversarial nodes.

1.1. Our contribution

In this paper, we introduce *Alphacodes*, a paradigm using visual puzzles that can be used in a variety of scenarios to enable a user to prove and verify the authenticity of critical portions of a transaction. Our key contribution is the design of a simple, easy to use protocol that enables users to verify if the transactions performed are secure, ergo providing them greater confidence and satisfaction. This protocol is without any complex cryptographic operations, and is over clear-text channels with potentially untrusted providers.

Alphacodes are a set of simple code-tables/books (or visual puzzles) shared between various interacting parties. Every time one party wants to conduct a transaction, he performs the encoding of specific parameters using the simple task of pattern-matching. The transaction succeeds only if the counterparty, upon decoding the messages of the transaction, meets certain constraints. 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. Essentially, each Alphacode is a spin-off of the popular Message Authentication Code (MAC) paradigm. However, unlike conventional MACs, Alphacodes are reversible and produce output of length of the same order of magnitude as the input. In essence, Alphacodes can be viewed as one version of short verification codes (replacement for randomly generated confirmation codes). Unlike traditional authentication mechanisms which primarily just au-

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

thenticate identities, Alphacodes can be used for authenticating the important transaction digits in each transaction.

We demonstrate how the Alphacodes paradigm is secure against MitM attacks, replay attacks, spoofing (cellphone or SIM) and prove its security guarantees. The simplicity of the attacks described coupled with the ease of launching them urges us to rethink the existing security primitives. The ease of generation and solvability of Alphacodes makes it a viable solution, 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. 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. PROBLEM CONTEXT

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

Branchless Banking: According to a study by CGAP and the World Microfinance Forum on financial inclusion, the leading state-owned banks including the Postal Savings Bank of China and the Agricultural bank of China boast unthinkable numbers of retail bank accounts, at 475 million and 320 million, respectively. One out of every three Chinese citizens holds 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 poorer families, particularly those in remote areas, have trouble accessing accounts and use them mainly for encashment, which can often require costly travel to bank branches or ATM in distant cities and towns. Thus, the poor have to ultimately turn to local merchants for loans 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. It makes no economic sense for a bank to have a branch in rural areas as the value of deposits are low and the people have poor or non-existent credit histories. 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 [9; 19]. People are not certain whether it is a safe way to conduct financial transactions [11]. 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.

Money transfer and Remittances: A significant fraction of families in developing regions rely on remittances from household members 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 interaction step of the money transfer involves the user revealing most of the sensitive information attributes to the agent for completing the transaction. While, one may envision SMS-based mobile banking to have enjoyed wide acceptance in developing nations such as Kenya, Tanzania, Phillipines and India, the ground reality is that the user is still dependent on the agent to procure the money. For example, M-Pesa in Kenya has adopted over 85,000 agents throughout the country who form the backbone of the mobile banking industry. The security in SMS based mobile banking is minimal with several known instances of security breaches [19]. We note that in the case of mobile banking, encryption does not solve the last hop human-provider interaction problem where the provider may not be trustworthy.

There has been very limited related work in trying to secure branchless banking and money transfer applications. The work by Sharma et al. [22] uses simple nonces among the user, agent and bank to provide secure protocols for deposit and withdrawal in a rural setting where a farmer and a shopkeeper use the shopkeeper's insecure phone to access 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 code-books, as the user interacts with the bank [20]. Unfortunately, as the authors note [20], their authentication scheme does not guarantee strong security again st adversaries who can spoof caller IDs or mount MitM attacks. The concept of code-books 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.

3. ALPHACODES

In this section, we present our straw-man to alleviate the problems discussed in §1. To briefly summarize, we generate an authentication (verification) code using physical code-books populated with sufficiently random entries. This authentication code coupled with transaction details sent in the clear can be used to ensure both transaction integrity and authentication. Consider Alice and Bob who wish to transact in a secure manner. They pre-share a pair of Alphacodes, using a secure physical channel such as secure courier. By encoding a number using the Alphacode, Alice can send Bob the information securely.

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

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. This is done as follows:

1. The first column in Table 1 are the numbers which Alice chooses in order and matches with the corresponding letters in the subsequent columns. To elaborate, 2 corresponds to codeword HY in the first codeword column, 5 corresponds to CA (in the next column) and 0 corresponds to CK (in the rightmost column). Therefore, 250 is encoded as *HYCACK*.

2. Alice sends *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 characters per digit, the probability of generating another valid sequence (collision) is 26^{-kn}. It is also interesting to note that for N entities communicating in a peer-to-peer fashion, we would require N^2 tables. In subsequent sections, we explain how we can reduce this number.

3. Similarly, Bob encodes 250 using Table 2 obtaining *KJGTRF*. He sends the encoding 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 this encoding, and responding with the correct response is negligible.

Note that throughout the paper, we use the term encoding to refer to the process of generating the appropriate verification code based on an input. Similarly, decoding refers to the reverse process. In preferred usage, the tables are used for one transaction only, ensuring uniqueness. To perform another transaction, another set of secret tables have to be shared 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 supports both message authentication and integrity. Only the possessors of the code-books 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.
— *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, yet their user tests indicate that even illiterate users can do this.
— *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.
— *Offline*: Alphacodes can be distributed in an offline fashion. For example, the bank could provide paper copies of Alphacodes. The customers could go to the bank and get the Alphacodes (in a check book format) or the bank could mail the them to the customers. The customer can then login to the bank's site using the Alphacodes which would be known only to the bank and the user.

4. ALPHACODE PROTOCOLS

In previous sections, we observed that peer-to-peer communication involved Alphacodes which were quadratic in the number of interacting parties. Explained because of its simple nature, this variant of the protocol does not scale well. In this section we introduce a three-party protocol which require code-books that scale linearly in the number of interacting parties. These parties interact in a secure manner based on their mutual trust and supervision of a trusted authority. For consistency with previously used nomenclature, we label the actors: user, service provider, and trusted authority. Explaining the setup once more, the user wishes to conduct a transaction with a service provider that may not be trusted, and hence wishes to ensure that his personal or sensitive information is not misused. The trusted authority checks the personal details and sensitive information and either commits or aborts the transaction between the user and the service provider. We make the following assumptions with respect to the actors and protocol:

(1) The user and trusted authority behave in an honest manner. Any deviation from this behavior will produce unfavorable results for both.
(2) The communication channel between the authority and the service provider is secure and trustworthy in one direction.
(3) The communication channel between the user and the authority 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.

4.1. 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 our initial assumptions, each service provider is issued 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. To briefly summarize the working of the paradigm, 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 preferentially states the amount of the transaction, though it could also encode approval (commit) or disapproval (abort) of the transaction. The protocols concern an amount/quantity that could be in some units of currency or weight or some form of numerical identifier. If the Alphacode allowed other symbols as input characters, then any set of those symbols could be used. 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 $\{ UT1_1, UT1_2, \ldots, UT1_n \}$, as well as $\{ UT2_1, UT2_2, \ldots, UT2_n \}$. Similarly, the booklet provided to S will contain tables$\{ ST1_1, ST1_2, \ldots, ST1_m \}$ and $\{ ST2_1, ST2_2, \ldots, ST2_m \}$. The first set of tables are used for encoding transaction details whilst the second set of tables 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 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 courier. Alternatively, the user could pick 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.

4.2. 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) It is important to note that all three parties communicate with each other at some part of time. This is required for successful authentication of transaction details. 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. 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 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 *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

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

then $\delta_i = YTTUDONQ$.

Observe that the input for the code-book 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 *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*.

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

(2) Note 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. Thus, the amount of state information required grows linearly with the number of transactions authorized (or rejected) by the authority. We argue that this state information will not bloat up, given the nature of the use-case for the Alphacodes paradigm in ensuring satisfaction within a semi-literate populous who transact infrequently, 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 code-book 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, based on state information, A gets the corresponding corresponding $ST1_j$. 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. Those two zeros would be transformed into four letters, signaling 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 (re-transmission), 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 from U to S.

S :	Compute $\beta'_j = alphacode(Am, ST2_j)$
U :	Compute $\alpha'_i = alphacode(Am, UT2_i)$
If (for U) :	$\alpha_i \neq \alpha'_i$
	Transaction has been aborted by A
If (for S) :	$\beta_j \neq \beta'_j$
	Transaction has been aborted by A
If (at U) :	$\alpha_i = \alpha'_i$
	U knows that transaction
	has been committed by A
If (at S) :	$\beta_j = \beta'_j$
	S knows that transaction
	has been committed by A

(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 response is not well formed based on the Alphacode (i.e. the encoded text does not correspond to any possible sequence of code-words), then the party receiving such a malformed message simply requests for a re-transmission. 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.

4.3. 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.

4.4. Analysis

In this subsection, we wish to highlight some of the salient features from the above discussion. The most basic property of Alphacodes can be stated by the following simple lemma:

LEMMA 4.1.
For an Alphacode using the English alphabet, the probability of guessing each entry of an English-based code-table correctly is $26^{-\ell}$, where ℓ is the size of the Alphacode for a given transaction input.

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 guess an entry correctly 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 for the same size of the input entry.

We highlight the following key properties of the protocol:

(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 implies adversarial presence. In this case, A computes α_i and β_j using zeros as the parameter in operation *alphacode* and aborts the transaction. This is required as if A sent $ABORT$ or any other message in cleartext, the adversary might be able to change it and continue the transaction. It is hard for an adversary to decode either α_i or β_j, or forge a message from A having zeros because the adversary can encode zeros properly with negligible probability without the Alphacode table. Denial of transaction attacks are dealt with in §4.3.
(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 uses possible outputs 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 is not composed of possible outputs, 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.
(4) If $\alpha_i = \alpha'_i$ then U knows that α_i is authentic and it was sent by A and that U and S agreed on the amount and that A is aware of this amount. If $\beta_j = \beta'_j$ then S knows that β_i is authentic and it was sent by A. It also means A has certified that the amount it receives agrees with the amount that U and S agreed to and that in fact U and S did send the same amount.

5. THWARTING ATTACK VECTORS

GCASH, M-Pesa, mChek, Obopay, Zain, Cellpay are some of the well known players in the field of mobile banking. A paper on mobile banking technology by Ignacio Mas of CGAP [2; 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). 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.

5.1. Replay Attacks

If the service provided is based on SMS, replay or outright forgery attacks are easy to facilitate as most of the SMS in the developing world uses A5/0 or A5/2 algorithm for over-the-air encryption. A5/0 is a dummy encryption and the 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 A5/2 is a weaker form of A5 encryption that has been phased out by the GSM Association, but is still in widespread use in developing regions. A5/2 can be decrypted in real-time [5; 25] and this makes SMS using A5/2 encryption insecure. 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. Alphacode protocols, on the other hand, thwart replay attacks as Alphacodes cannot be reused to perform a new transaction. The trusted authority detects reuse but does not repeat the transaction as explained in §4.

5.2. Spoofing and Phishing

Though new SIM cards (Version 3) are harder to clone, older versions of SIM cards can be cloned [24]. 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, oblivious to 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 different secure channel, to complete the transaction.

5.3. Man-in-the-Middle Attacks

In mobile banking, one can launch a MitM attack by configuring a USRP radio to act as a fake base station [19; 3]. In such an attack, nearby cellphones associate themselves to the USRP device instead of the actual base station. Each handset uses the GSM mandated ciphering algorithm, but the base station can negotiate the ciphering algorithm and bring it down to A5/0 (i.e. no encryption). Once a cellphone associates itself to a USRP device, the transaction details that pass through the USRP could be modified to suit the adversary unwitting to the user and the service provider. We have shown that the Alphacode protocols are secure against MitM attacks in §4.4.

5.4. Rootkit

Rootkits are malware that stealthily modify the OS functionality to achieve malicious goals. Cabir, Mabir, Skull.D [1] are some of the popular malware that infected the cellphones through SMS, MMS or Bluetooth. Bickford et. al. [4], have shown that it is possible to eavesdrop on cellphone voice conversations if the cellphone is infected with a rootkit. This has serious consequences to Interactive Voice Response (IVR) based banking services such as mChek. The Alphacode protocols are secure against rootkit attacks, as the probability of the protocol to succeed if there is any type of interception and modification of transaction details is negligible as shown in §4.

6. APPLICATIONS

This section explains how the Alphacode protocols can be applied in many real-world scenarios.

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 §4. Transactions made by the purchaser are validated by the bank or credit card provider in the absence of any malicious activity from the vendor.

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 Alphacodes with an end-user through a trusted channel such as a physical channel or signed e-mail. Using the Alphacode, the user follows the steps in the §4 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 a 140-byte SMS message.

6.3. Voice-based Transactions and Phone Banking

We assume that phone lines constitute an unsecure 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 §4. This illustrates that the protocol works regardless of medium (text or voice).

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 §4 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. This assignment for deposit and withdrawal is known to both the farmer and the shopkeeper. 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. IMPLEMENTATION

We have implemented two simple applications to enable secure transactions using the Alphacodes framework.

E-commerce transactions on the web: We assume that the user who aims to perform E-commerce transactions registers with a trusted authority which is responsible for issuing an unique identity to the user and there is secure channel for the trusted authority to disseminate Alphacodes to the user. Here, the Alphacode puzzles are similar to the use of per-transaction CVV codes or PIN numbers used in credit cards. We also assume that every Alphacode is associated with a unique puzzle number apart from the actual codes. Assume that a user performs an E-commerce transaction on an untrusted site. We created a simple E-commerce site which simulates an online shopping site where the user can select several items in a shopping cart and perform an online transaction. When completing the transaction, the user has to enter the user identity, the trusted authority issuing the identity and Alphacodes, puzzle number corresponding to a chosen Alphacode and the solution to the sender puzzle based on the significant digits of the transaction amount. The untrusted E-commerce site needs to obtain authorization for the transaction amount from the trusted authority and hence needs to share the user information along with the Alphacode puzzle number and the sender code. The trusted authority can accept or reject the transaction depending on the whether the solution to the Alphacode is correct. If correct, the trusted authority shares the recipient code response back to the untrusted site which needs to be disseminated to the user as the confirmation code for the transaction. If the confirmation code does not match the correct trusted authority response, the user can initiate a dispute resolution for the transaction with the trusted authority.

SMS Money transfer Application: Here, we assume that all users have a unique identity issued by a bank. The user identity is also directly linked to the mobile number of the user. The trusted authority 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 the trusted authority. 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 SMS mobile number corresponding to the bank. The bank checks the code and if its correct sends the corresponding response code over SMS to the recipient. If the response code does not match the expected response, the sender can initiate a dispute resolution with the bank.

8. MECHANICAL TURK EXPERIMENTS

We conducted two experiments on Amazon Mechanical Turk to find the ease of solvability of Alphacodes at a user-level. These tests are necessary to understand the ease of adoption of the paradigm by a target population.

8.1. Sample

As per guidelines set by Amazon, all the participants were from the Unites States, and are naturally assumed to have working knowledge of the English language. These participants (referred to as workers) are screened by Amazon and expected to follow a set of guidelines ensuring honest and quality responses to tasks raised by requesters. Hence, we believe that this should be deemed 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 both the crowd-sourced experiments, two solved questions were provided at the beginning of each task as examples. In essence, users had to bootstrap the Alphacodes concept using only two examples.

8.2. Task Definition

As shown in the examples prior, each Alphacode table (or grid) was composed of 10 rows and K columns, where a K digit number was provided as input. We use $K = 4$ 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 specifically confuse the users and were set as slight variants of the correct answer. We expected 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 workers are motivated to act honestly due to strict regulations enforced by

Amazon on deviants. In subsequent subsections, we describe the experiments in greater detail and explain the results obtained.

8.3. 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 randomly chosen 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 content populating the grids, and 10 had Non-English (or symbolic) content. 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. The average time taken to solve the puzzles indicates the simplicity of the paradigm proposed, suggesting easy adoption among the semi-literate masses. As the users were not all provided the same puzzle, we are unable to comment on the difficulty of each puzzle individually. 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 with knowledge that the participants understood how to use Alphacodes with limited training.

8.4. 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, performing a function similar to that of a one-time pad. 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 0's 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 turk 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 to solve all 20 transactions.

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). 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. Unlike the first experiment, the sender puzzle error rate was marginally higher since we expected the users to type down the answer 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. We also believe that performing the required operations at an application level will further reduce the error to a negligible value

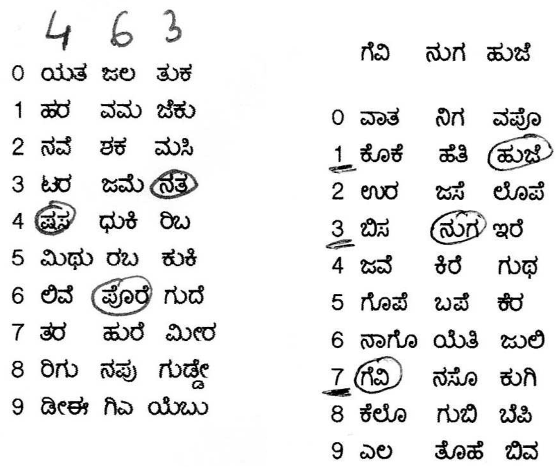
9. ALPHACODES SOLVABILITY IN A LOCAL LANGUAGE

To demonstrate the power of people to solve Alphacodes in a local language, we perform a simple user study in the Kannada language (Indian language in the state of Karnataka) with a small population of 10 users who all spoke the language. We gave the participants two tasks to complete. In the first task, the participants were given a sheet of paper which consisted of an Alphacodes table. In the table, the English letters were replaced with the letters of the local language, 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 instructed of the rules and a few sample tables were filled by us to show the participants the rules of the process. The teaching part took about 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. The rest needed more time to understand the process, but they too completed the task after under going additional training.

The second task was as follows: A set of letters were written at the top of the table and the users had to match the letters 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. Note that Kannada is significantly more complex than English with over 500 different written symbols in the alphabet. 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 were able to satisfactorily complete the task as per the specified procedure. We believe that the training in the first task helped the users in more quickly grasping the concept for the second task.

Figure 1a 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 1b shows an example table



	4	6	3
0	ಯತ	ಜಲ	ತುಕ
1	ಹರ	ವಮ	ಜೆಕು
2	ನವೆ	ಶಕ	ಮಸಿ
3	ಟರ	ಜಮೆ	[✓ ನತ]
4	[✓ ಷಸ]	ಧುಕಿ	ರಿಬ
5	ಮಿಥು	ರಬ	ಕುಕಿ
6	ಲಿವೆ	[✓ ಪೊರೆ]	ಗುದೆ
7	ತರ	ಹುರೆ	ಮೀರ
8	ರಿಗು	ನಪು	ಗುಡ್ಡೇ
9	ಡೀಈ	ಗಿಎ	ಯೆಬು

(a)

	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	ಯೆತಿ	ಜುಲಿ
7	[✓ ಗೆವಿ]	ನಸೊ	ಕುಗಿ
8	ಕೆಲೊ	ಗುಬಿ	ಬೆಪಿ
9	ಎಲ	ತೊಹೆ	ಬಿವ

(b)

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

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.

Both these simple experiments demonstrate two powerful facts about Alphacodes: (a) The concept of Alphacodes can be easily extended to other languages beyond English quite easily; (b) The time taken for novice users to learn about the Alphacodes concept is quite small. Given repetitive practice, users can quickly master solving Alphacodes.

10. ADDITIONAL 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 [23]. Our paper shows 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 [8] 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 [7] propose a protocol for sharing secret information between two parties using a deck of cards to play discreet solitary games.

Scheiner [21] 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 [6] 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 understand and use on a daily basis.

Two-factor authentication is being used by a number of organizations to provide extra security to users [10]. Although it provides an additional layer of security, it does not take away 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.

11. 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, due to the convenience it provides the users. This convenience comes at the risk of exposing the users to different forms of MitM attacks for several types of real-world commerce transactions. This paper aims to build new protocols based on the concept of Alphacodes which are simple to use, secure against failures in infrastructure, and should prevent exploitation of the poor by unscrupulous merchants. Alphacode protocols are simple to use and can specifically enable new types of secure transactions relevant in developing regions. The Alphacode protocols are designed to tolerate dropped messages, corrupt messages, man-in-the-middle attacks, and other malicious attacks. We believe Alphacodes represent a very simple and effective way of bootstrapping security in end-to-end transactions especially in phone-based transactions, untrusted e-commerce site, remittances, mobile banking and branchless banking contexts.

REFERENCES

F-secure. http://www.f-secure.com.

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

Practical Cellphone Spying. https://www.defcon.org/html/defcon-18/dc-18-speakers.html#Paget.

Bickford, J., O'Hare, R., Baliga, A., Ganapathy, V., and Iftode, L. Rootkits on smart phones: attacks, implications and opportunities. HotMobile '10, ACM, pp. 49–54.

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

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

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

Fagin, R., Naor, M., and Winkler, P. Comparing information without leaking it. *Commun. ACM 39* (May 1996), 77–85.

GCash, G. http://www.234next.com/csp/cms/sites/
Next/Home/5413169-146/Branchless_
banking:_Uncertainties_emerge.csp.

Grosse, E., and Upadhyay, M. Authentication at scale. *IEEE Security and Privacy 11* (2013), 15–22.

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

Lord, S. Trouble at the Telco: When GSM goes bad. *Network Security* (January 2003).

Mas, I. Economics of Branchless Banking. *Innovations Vol. 4, Issue 2* (January 2009).

Moran, T., and Naor, M. Basing cryptographic protocols on tamper-evident seals. In *ICALP 2005* (July 2005), L. C. et al., Ed., vol. 3580 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 285–297.

Moran, T., and Naor, M. Polling with physical envelopes: *a* rigorous analysis of a human-centric protocol. In *Eurocrypt 2006* (May 2006), S. Vaudenay, Ed., vol. 4004 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 88–108.

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

Naor, M., and Shamir, A. Visual cryptography. Springer-Verlag, pp. 1–12.

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

Paik, M. 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* (New York, NY, USA, 2010), HotMobile '10, ACM, pp. 54–59.

Panjwani, S., and Cutrell, E. Usable, low-cost, authentication for mobile banking. In *SOUPS* (2010).

Schneier, B. The solitaire encryption algorithm: http://www.schneier.com/solitaire.html.

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

Stinson, D. R. Universal hashing and authentication codes. In *International Crytology Conference*, pp. 74–85.

Wagner, D., and Goldberg, I. GSM Cloning. http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html, 1998.

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