Towards End-to-End Security in Branchless Banking

Saurabh Panjwani
Microsoft Research India
saurabh.panjwani@microsoft.com

ABSTRACT

Mobile-based branchless banking has become one of the key mechanisms for extending financial services to disenfranchised populations in the world's developing regions. One shortcoming of today's branchless banking systems is that they rely largely on network-layer services for securing transactions and do not implement any application-layer security. Recent attacks on some of the most popular branchless banking systems show that these systems are *not* end-to-end secure.

In this paper, we make the case for designing mobile-based branchless banking systems which build security into the application layer of the protocol and guarantee end-to-end security to system users. Our main contribution is a threat model which effectively captures the goals of end-to-end authenticated transactions in branchless banking. This model, besides incorporating the obvious external threats to a protocol, also accounts for the possibility of *insider* attacks—those mountable by banking agents or other human intermediaries in the system. We then provide recommendations for solution design based on the security requirements of our model and the infrastructural constraints under which branchless banking systems operate.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Security and Protection; K.4 [**Computers and Society**]: Electronic Commerce

Keywords

Branchless banking, security, mobile networks, GSM

1. INTRODUCTION

While western societies continue to advance and experience the benefits of modern financial services, there remain 2.5 billion people in the world without access to a basic savings account [7]. Recent efforts, under the hood of *branchless banking*, are beginning to bridge this gap by enabling banks and other financial institutions to operate in remote areas without incurring the cost of setting up brick-and-mortar offices or ATM terminals. Systems designed under this paradigm operate through a network of human *agents* who facilitate cash-based transactions for beneficiaries and use cheap and prevalent technologies for communicating with the service provider. There are tens of branchless banking systems already in existence today and in sum, they serve more than 50 million people and transact over $100 million on a daily basis [11, 10].

The deep penetration of mobile phones in developing regions makes cellular networks the preferred communication medium in branchless banking. Transactions typically involve a customer and an agent meeting in one location, sending and receiving messages to/from a bank server through their mobile phones and transferring some physical cash from one person to the other. As in any financial service, ensuring security of transactions is paramount and indeed, most of the current branchless banking systems take steps in this direction.

One shortcoming of most of today's branchless banking systems is that they provide security at the network layer only and do not implement any application-layer cryptography. For example, M-Pesa [5], which is the pioneer of the branchless banking concept and serves over 50% of Kenya's adult population, uses a custom-made SIM Tool Kit (STK) program to protect transaction messages exchanged between client phones and the server. Not much is publicly known about M-Pesa's security algorithm but recent attacks on the system reveal that it does not guarantee end-to-end security to customers [17]. Other key players like G-Cash [3] in the Philippines rely directly on GSM's default security services to protect client information but these services are known to offer very weak security guarantees [9, 19]; in [13], it is argued that vulnerabilities in GSM's security suite could be used to subvert G-Cash transactions. Indeed, the question of *what* application-level security means in the context of branchless banking does not seem to be well-understood yet, neither in the academic literature nor in practice.

In this paper, we take the first step in addressing the above question. We present a protocol and threat model which captures the goals of end-to-end security in branchless banking systems. We focus on the task of *authenticating* transactions in this paper; ensuring transaction privacy in branch-

An abridged version of this paper appears in the *12th Workshop on Mobile Computing Systems and Applications (HotMobile) '11*, March 2011. This is a more complete version. Alternate email address of the author: saurabh.panjwani@gmail.com. Please use this for correspondence if the above address does not work.

less banking, as will become evident, is fairly non-trivial and will be explored in future work. Our threat model, besides incorporating the obvious external threats on protocols (e.g., man-in-the-middle tampering), also addresses the possibility of *insider* attacks—attacks in which an agent or a customer behaves maliciously against other parties. For example, we assume that agents, who are proximally present during transactions, can eavesdrop on information provided by customers and can use this information adversarially.

We then proceed towards designing a system that meets the security requirement of our model. There are two unique challenges to this design task, which we highlight here. The first is that of insider attacks: how do customers authenticate themselves to the bank given that their credentials could be easily leaked to nearby agents? Knowledge-based credentials like passwords are a weak candidate tool for this because they operate well only in scenarios where eavesdropping is not feasible. The use of biometrics, though an interesting possibility, is fraught with issues of privacy and irrevocability, which are exacerbated in the current context by the eavesdropping power available to bank agents. We thus propose the use of one-time password (OTP) based hardware tokens in our system since they seem to provide the best defense against eavesdropping and are also commonly used in practice [4]. Though OTP-based schemes have some limitations (e.g., OTP tokens are susceptible to theft), we discuss how these limitations could be addressed by carefully augmenting OTP's with the use of fixed numeric passwords.

The other issue which makes our task difficult is the context in which branchless banking operates. These systems cater to some of the world's poorest people, who have limited educational backgrounds and who rarely possess programmable phones. (Over 100 million people in India still seem to own a dumb phone [18].) The implications for design are: (a) use a simple interface (minimize protocol steps and human computation requirements); and (b) to the extent possible, avoid programming client phones. While we are able to reasonably satisfy the first constraint, we require at least some phones in our system to be programmed for without this, end-to-end security seems impossible to achieve. We first present our ideas to build systems assuming *all* phones are programmable. Later, we present variations for settings where only *agent phones* are assumed programmable. These variations make additional setup assumptions (e.g., greater use of OTPs) and achieve slightly-relaxed forms of security. We note that a few existing systems already assume programmability of agent phones [1], an indicator that the assumption may not be a deterrent to deployment.

2. RELATED WORK

Despite the rife deployment of branchless banking systems across the world, research on security issues in these systems is extremely sparse. Most of the currently-deployed systems seem to either rely on GSM's inbuilt mechanisms for secure transactions while some develop their own network-layer schemes to do the same. The pitfalls of network-layer security design are well-documented in the literature [15] and relying on the inbuilt security mechanisms of 2G GSM, the most commonly used communication standard in branchless banking, is particularly risky given the long history of security exploits associated with this standard [9, 19].

An important case in point is M-Pesa [5], perhaps the biggest and most cited branchless banking system in the world today. In this system, each client (customer or agent) authenticates himself to the bank during transactions using a 4-digit PIN uniquely assigned to him. (The PIN is sent via USSD along with other transaction details.) The SIM card of the client is pre-programmed with cryptographic software which purportedly protects the client's PIN during transit. The algorithm implemented by this software is not publicly known. While such practice of obscuring algorithmic details of cryptographic software is in itself questionable, we point out two other limitations of the M-Pesa solution. First, the solution exists entirely at the network layer, which implies limited protection, if any, from network operators (e.g., network operators could have complete access to client PINs). Second, protecting client PINs alone does not necessarily imply that the system is end-to-end secure. Indeed, a recent attack on the system [17] confirms that the solution does *not* provide end-to-end security to clients, neither to customers nor to agents. In this attack, a malicious customer, with the help of a remote conspirator spoofing SMS's on behalf of the bank, was able to convince an unsuspecting agent to yield cash even without the bank having recorded a cash transfer. The attack exploited the simple fact that the system does not enable clients to authenticate bank-originating SMS's, thus making them susceptible to easy spoofing attacks [6].

Some branchless banking systems, like that of [1], do seem to implement application-layer security protocols but the protocols, again, are not publicly known and evidence for end-to-end security is weak. For example, the solution of [1] uses fingerprint biometrics to authenticate clients but seems to assume honest behaviour of all agents: agents are assumed incapable of acquiring and misusing user credentials. Our goal, on the other hand, is to design a solution that admits the possibility of malicious clients. We desire that customers be guaranteed a high level of security even when transacting through eavesdropping agents.

In [16], a system for secure branchless banking based on OTP tokens and voice biometrics is proposed, but this system is not end-to-end secure: it does not enable agents to authenticate bank-originating messages (leaving them susceptible to spoofing attacks) and it assumes that GSM voice calls are not interceptible, an assumption that has recently been nullified—with a *feasible* attack—by mobile security researchers [19]. (Previous interception attacks on GSM were too expensive to mount.) Protocols also have high round complexity—six message transmissions per transaction. Finally, the deployability concerns of voice biometrics are left unresolved. We hope to develop a scheme that is not only more secure than that of [16], but is also simpler, more efficient and capable of being deployed in developing regions.

3. SYSTEM MODEL AND THREATS

We present a model for branchless banking systems and the threats they could be subjected to in practice. Our system model is an abstraction of well-known branchless banking implementations and systems like M-Pesa [5], G-Cash [3] and Eko [2] which operate in Kenya, Philippines and India respectively are instances of it.

3.1 Basics

A branchless banking system involves three types of entities: *bank*, *customer* and *agent*. There is a single bank B in the system and all customers and agents are its *clients*. The bank manages financial information for all clients[1]. Agents are special clients who assist customers in conducting financial transactions with the bank.

Clients engage in different types of protocols and each protocol involves remote communication with the bank. We assume that every client X in the system owns a personal digital device, denoted $\mathcal{D}_X$, which can communicate with a server maintained by the bank. Since wireless technologies like GSM have the deepest penetration in developing regions, in practice these devices will likely be mobile phones capable of communicating via a GSM (or similarly-standardized) interface. Although the condition of every *customer* owning a device is not essential (one device per agent suffices), most deployed systems assume it and the assumption leads to greater functionality and security.

3.2 Protocol Types

To join a branchless banking system, each client X must execute an *enrolment* protocol wherein he submits identification information to the bank (e.g., ration card, proof of residence) and the bank, after carrying out suitable background checks, assigns and communicates to him a unique account number ID_X. (This could be just a unique identifier of X's personal device $\mathcal{D}_X$, as is the case in some existing schemes.) The bank server maintains a record for each enrolled client in a database $\mathcal{T}$. When a client X enrolls, an entry for X is created in $\mathcal{T}$ and the entry includes, among other things, the value ID_X and the account balance of X, $\texttt{bal}(X)$, initially set to 0. The enrolment protocol is executed either in person or, when this is infeasible, using other reliable and trusted communication means (e.g., a courier service). Enrolment is more elaborate for agents than for customers.

Enrolled customers can carry out deposit and withdrawal transactions just like in any banking system but here, all such transactions are mediated by agents. Every agent A in the system is required to maintain a cash float of a designated minimum value $\texttt{m}$. A *deposit* protocol involves a 3-way interaction between a customer, an agent and the bank: if C wishes to deposit cash of value x into his account, he visits a nearby agent A and makes a transaction request. A then sends the information $(\texttt{d}, ID_A, ID_C, x)$ to the bank server, along with her credentials, using device $\mathcal{D}_A$. The bank increments C's balance $\texttt{bal}(C)$ by x and simultaneously *decrements* $\texttt{bal}(A)$ by x; this signifies movement of virtual currency from A's account into C's. The bank then replies with a confirmation to $\mathcal{D}_A$ and $\mathcal{D}_C$. Upon viewing the confirmation, C gives real currency (of value x) to A.

A *withdrawal* protocol is a near-converse of the above: to withdraw cash of value y from his account, C visits an agent A' (could be the same as A), makes a transaction request but this time *he* sends the required information $(\texttt{w}, ID_C, ID_{A'}, y)$ to the bank, through $\mathcal{D}_C$ or $\mathcal{D}_{A'}$. C also includes his credentials in the withdrawal message in order to authorize the bank to reduce his account balance. If $\texttt{bal}(C) \geq y$, the bank decrements $\texttt{bal}(C)$ by y and simultaneously increments $\texttt{bal}(A')$ by y (signifying virtual money transfer from C to A') and sends a confirmation to $\mathcal{D}_{A'}$ and $\mathcal{D}_C$. (Otherwise, an error message is sent.) C receives the corresponding cash from A' if and only if the confirmation is received.

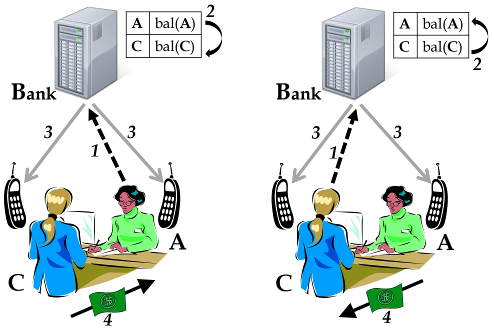

Figure 1: A deposit protocol (left) and a withdrawal protocol (right) involving customer C and agent A. Arrows are numbered based on the order of protocol processes: (1) send transaction request to the bank (through $\mathcal{D}_A$ or $\mathcal{D}_C$), (2) bank changes balances, (3) bank sends confirmation, (4) transfer cash.

Clients can view their current balance at any point using a *balance check* protocol: to check his balance, customer C sends a request to the bank server using $\mathcal{D}_C$ and receives the value of $\texttt{bal}(C)$ in return. Some systems also implement a *transfer* protocol using which customers can move virtual money from their accounts into other customers' accounts. Such a protocol requires the money sender to submit his/her credentials.

From time to time, agents transfer cash in excess of $\texttt{m}$ to the bank (through an in-person visit) and for each such transfer of value x by an agent A, the bank increments $\texttt{bal}(A)$ by x. If the cash float of A ever falls below $\texttt{m}$ in value, it is replenished by the bank and $\texttt{bal}(A)$ modified suitably. Agents receive commissions on a per transaction basis, and could even be partly or fully employed by the bank.

3.3 External Threats

We consider two types of attacks on a branchless banking system—external and internal. External attacks are those which disrupt communication between system entities. We assume the existence of an adversary who can eavesdrop on all communication between client devices and the bank server, drop messages selectively and, possibly, inject spoofed messages (i.e., messages with arbitrary sender addresses) into the channel. Giving the adversary full eavesdropping privilege is justified given known vulnerabilities in encryption schemes of current-day mobile networks [9] and the fact that these schemes are applied differentially on transmitted messages; e.g., GSM's encryption functions operates only on the traffic channel (if at all) and leaves messages on the signaling channel, including SMS and USSD messages, unencrypted. Furthermore, in certain messaging protocols on mobile networks, spoofing sender identities is also a practicable threat; e.g., spoofing the originating address of an SMS is known to be easy [6].

Besides eavesdropping on and spoofing messages, we assume

[1] We use the term "bank" loosely here, referring essentially to the entity that provides the branchless banking service. In implementations, this entity may not be a formal bank.

that the adversary has the ability to mount man-in-the-middle attacks on the channel. In particular, he can intercept any message sent from an agent or customer device, modify it and re-transmit the modified version to the bank server. Such an attack model is justified by recent experiments in mobile telephony which demonstrate the feasibility of impersonating GSM base stations to mobile devices with fairly modest resources [19]. Not only that, base station impersonation attacks in GSM networks can force mobile devices into disabling the use of network encryption, which makes man-in-the-middle attacks even more threatening.

3.4 Internal Threats

It is plausible that internal parties in a branchless banking system are sufficiently incentivized to play malice against each other. In particular, since agents facilitate transactions for multiple customers (of the order of hundreds sometimes) and have a relatively limited amount of control exercised over them (owing to their physical distance from the bank's premises), they could easily be inclined to deviate from prescribed behaviour. Similarly, customers, being remote from the bank, could try to violate norms during transactions and cheat either on other customers or on agents.

We thus allow certain adversarial powers to all parties (except the bank) in a branchless banking system. We assume that agents have two types of capabilities: first, they can *eavesdrop* on any information that is provided by customers during transactions. This captures the possibility of physically observing such information (e.g., shoulder-surfing of PINs) as well as of acquiring it using electronic tools (like biometric skimmers and key loggers). Second, we allow agents to *tamper* with software on their respective devices; so, for example, agent A can modify an application provided by the bank on $\mathcal{D}_A$ and cause it to replace messages of the form $(\texttt{w}, ID_C, ID_A, y)$ with $(\texttt{w}, ID_C, ID_A, y + \Delta)$ while still displaying legitimate confirmations on the display. Eavesdropping and software tampering attacks seem more feasible in branchless banking than in ATM-based banking due to the proximal presence of agents in every transaction.

We assume similar capabilities for customers as well: they can eavesdrop on information submitted by any agent they transact with (e.g., they could view an agent A's input on device $\mathcal{D}_A$ during a deposit) and can modify applications on their respective devices. Finally, we allow all parties in the system the power to acquire another party's device and to either tamper with it or execute banking protocols using it. This models the possibility of device theft, which is a security concern in any mobile-based application.

3.5 Security

We focus on the security of withdrawal and deposit protocols in a branchless banking system. For security, we require that whenever an agent A and a customer C complete a withdrawal/deposit protocol and at least one of them is honest, then the view of the protocol from the perspective of each honest party must be exactly the same as the view from the perspective of the bank. For example, if C honestly completes a withdrawal protocol (using his own untampered device) and C believes that the protocol has decremented his balance by x and incremented A's balance by x, then the bank's database $\mathcal{T}$ must reflect exactly this result. We do not require privacy of protocol transcripts since this seems non-trivial to ensure in the presence of eavesdropping agents. Future work will present more formal security definitions for branchless banking, cover protocols other than deposits and withdrawals and consider stronger security notions; e.g., transcript privacy and security against collusion attacks are two issues we hope to address later on.

4. TOWARDS A SECURE SYSTEM

We present some initial ideas on how branchless banking systems that meet the security criteria of the above threat model could be designed. Our description is sketchy at this point; filling in all details is part of our ongoing work.

4.1 Assumptions

We assume all client devices are GSM-compatible mobile phones. Furthermore, we require that some of these phones are programmable and, in particular, capable of transporting information generated using cryptographic schemes; this refers to the ability to run a protocol like GPRS and support an application platform like J2ME. We consider two scenarios.

- **Scenario 1: All phones are programmable.** This is the more restrictive scenario for developing regions since non-programmable phones dominate the mobile phone market in such regions [18]. Still, we consider it since it simplifies system design considerably and requires no further design assumptions. We remark that there is ongoing work on extending the capabilities of non-programmable phones using cheap hardware accessories [8]; such accessories, if produced at scale, can greatly increase the feasibility of Scenario 1.

- **Scenario 2: Agent phones are programmable.** This scenario is the more achievable one. Since agents in branchless banking systems are typically better off than customers and are even compensated for their work, they are more likely to either own a programmable phone already or be convinced into acquiring one. Some banks may even be in a position to commission a programmable phone to each agent—the phone incurs a one-time setup cost to the bank which gets amortized across customers. Not surprisingly, there are some systems which operate under this scenario [1].

We do not consider a scenario in which none of the phones are assumed programmable i.e., where all phones are capable of only basic messaging protocols like SMS, USSD and voice. Although this is the most practicable scenario and is assumed by almost all existing systems, it is difficult to design fully secure systems in this scenario, given recent developments in the mobile security space [19].

4.2 The Choice of User Credentials

Since our model enables clients to eavesdrop on each other's inputs during transactions, the choice of credentials to use for authenticating clients is critical for secure system design. Passwords are a weak candidate authentication tool since once an eavesdropper learns a client's password he or she can trivially impersonate as that client and conduct fraud. Biometrics are less easy to forge than passwords and they have significant usability advantages which is why several banks, including branchless ones [1], do use them in practice.

However, we argue that biometrics, too, are an inappropriate user authentication tool in branchless banking. First, voice biometrics—though an attractive choice, given that our platform is mobile phones—are unsuitable because of the ease with which they can be duplicated and the difficulty of deploying them in the real world; deployment is particularly a challenge for developing regions where the incidence of noise pollution is high. Second, other types of biometrics like fingerprints and iris scans are presently expensive to implement, expensive enough that using bio-scanners on a per client basis (the most secure option) is infeasible. Third, if one deploys a single bio-scanner per *agent* (as done by [1]) and shares it across all customers who transact with that agent, the cost of deployment may become bearable but the security risks escalate: agents have immediate access to biometrics of all customers they transact with and this, along with the threat of software tampering, makes fraudulent use of customer biometrics easier than in the one-scanner-per-client setting.

We also note that by their very nature, biometrics are an irrevocable form of credential: there is a limit to the number of times a credential can be replaced once compromised[2]. Thus, unless we assume complete honesty of all agents in our system or tamper-proofness of scanning software (both difficult possibilities!), biometric-based authentication is a risky proposition in branchless banking.

We propose one-time passwords (OTPs) as the primary user authentication tool in branchless banking systems. At the time of enrolment, every client X would receive a list of OTPs $(s_1^X, \ldots, s_n^X)$, generated randomly and independently, for some large n from the bank; a copy of this list would also be maintained at the bank server. The i^{th} OTP s_i^X would be used for authenticating X in the i^{th} transaction initiated by X; the OTP would be appended to the transaction message X sends to the bank. Since each OTP is meant to be used only once, the threat from eavesdropping attacks would be drastically reduced. In implementations, OTPs could be stored on small electronic dongles with small displays [4] or in paper form. Care would be needed to ensure synchronized use of OTPs between client and server and to prevent guessing attacks. As is standard practice, 6-digit OTPs should provide sufficient security.

We remark that a branchless bank named Eko [2], with a client base of over 300,000, has already deployed OTP tokens for user authentication in practice. A security weakness in Eko's scheme and a potential fix are reported in [14]. (The fix was devised in joint work with Eko.) Even with the fix, the system seems vulnerable to spoofing attacks as launched against M-Pesa [17], although no real execution of such attacks is known.

4.3 Preventing Spoofing and Tampering

To address the issue of spoofing and message tampering attacks, we recommend the use of a standard cryptographic tool—digital signatures. Signatures are easier to deploy in scenario 1, so we consider that scenario first and discuss suitable alterations for scenario 2 in Section 4.4. One candidate solution would be to use a public-private key pair for the bank, say (pk_B, sk_B), and one for each agent in the system. All phones would be equipped with the public key of the bank pk_B and a signature verification program during enrolment. In addition, the phone of agent A would store the public-private key pair of A, (pk_A, sk_A), which would also be generated by the bank when enrolling A. Client-originating transaction messages would be sent through agent phones only; e.g., for a withdrawal transaction that customer C conducts with agent A, the former would enter all details, including credentials, into $\mathcal{D}_A$ and the device would sign the information under sk_A before transmitting it. Since A could be malicious and could potentially tamper with her phone's software, it is critical that the bank's *confirmation* messages are also signed and later verified by each client device; e.g., a confirmation message for a withdrawal request made by C must be signed under sk_B and $\mathcal{D}_C$ must verify the signature using pk_B before approving the message. In order to prevent messages from being replayed, a timestamp should accompany every message that is transmitted.

We illustrate the use of signatures, OTPs and timestamps through a sample deposit protocol. To deposit amount x, customer C approaches agent A and the latter enters the message $m := (\mathtt{d}, ID_C, ID_A, x, s_i^A)$ into the device $\mathcal{D}_A$ with s_i^A being the first unused entry on A's OTP list. The device transmits $(m, \mathtt{sig}_{sk_A}(m))$ to the bank server. Upon receipt of a message (m, τ) at time t, the bank server first parses m as $(\mathtt{d}, ID_C, ID_A, x, s_i^A)$, checks that s_i^A is the first unused OTP on A's list and that τ is a valid signature of m under sk_A. If all this is true, it sets $m' = (\mathtt{ok}, \mathtt{d}, ID_C, ID_A, x, t)$; else, it sets $m' = (\mathtt{fail}, \mathtt{d}, ID_C, ID_A, x, t)$. It sends the signed message $(m', \mathtt{sig}_{sk_B}(m'))$ to both $\mathcal{D}_C$ and $\mathcal{D}_A$. The client devices then verify that the message doesn't contain fail, that it has a fresh timestamp and that the signature on the message is verifiable using pk_B. Only if these conditions are met is a confirmation reported and the cash transferred from C to A; otherwise, the protocol is deemed a failure. When a protocol fails, participating clients must communicate the failure event to the bank using suitable out-of-band methods and request either a re-transmission of confirmation messages or a transaction rollback.

We stress that this is only a sample protocol and several efficiency improvements can be devised using standard cryptographic techniques; these will be discussed in future work.

4.4 Addressing Scenario 2

Scenario 2 is more challenging, and arguably the more interesting one, to tackle. In this scenario, customer phones cannot be programmed, which essentially means we need techniques other than digital signatures to authenticate messages sent by the bank. While it may not be possible to achieve foolproof security under such a constraint, our belief is that we could get pretty close without significantly hampering efficiency.

[2] There is much research on making biometrics revocation-capable, but none of it applies to scenarios where human agents maintain the biometric scanning device and have access to original user features. Building fingerprint scanners that are resistant to biometric duplication (i.e. those which can distinguish real fingers from fake ones) is another active area of research but it is unclear if such scanners are capable of resisting future, unforeseen finger faking techniques.

Relaxing the model. One approach would be to consider security in a slightly-relaxed threat model wherein we rule out attacks which are known to be relatively improbable in practice. For example, in our current model, the assumption that adversaries can spoof messages on behalf of other parties is realistic, but that they can launch man-in-the-middle (MITM) attacks *arbitrarily* is quite extreme. While GSM's design makes MITM attacks feasible for sessions originating from a mobile phone, doing the same when only the *recipient* is mobile, but the session originator is not, is still relatively difficult [19, 12]. If the bank negotiates a reliable SMS channel with its network operator for push messages and a secure TCP/IP link to connect to the operator's SMSC, the risk of MITM tampering on the messages it sends to customers is drastically reduced.

If we rule out MITM tampering on bank-originating messages, we mainly need to guard customers from message spoofing attacks. This could easily be achieved as follows: The bank provides to each customer C a list of *response* OTPs $(r_1^C, \ldots, r_n^C)$, generated randomly and independently of each other and of the s_i's that C already received. For every confirmation message targeted at C, the bank appends the first unused OTP in this list to the rest of the message and sends it via SMS to $\mathcal{D}_C$. C verifies authenticity of the SMS by checking that it ends in the first unused response OTP on his list. If this is not the case, the indication is that the SMS has been spoofed, in which case the protocol fails. Alternate ways to transmit response OTPs (e.g., using voice or USSD) could result in different grades of security. A thorough investigation of security-usability trade-offs for spoof-protection of mobile transaction receipts is an interesting stand-alone problem left open by this work.

Special-purpose signatures. Another possible approach to tackle scenario 2, though less easy to deploy, is to design new signature schemes for which signature verification is computationally cheap and can be directly performed by humans. Although we believe it is impossible to design humanly-verifiable digital signatures which provide reasonable security *and* work for *arbitrary* messages, in the context of branchless banking, where messages have limited variability, a satisfactory solution may very well exist. The main variables in a bank's confirmation message which need verification are the transaction amount and the transacting agent's identity and these can be represented with about 10 digits in most implementations. Furthermore, for a given customer, the number of possible values of these variables is often small enough to make them representable with even fewer digits. In ongoing work, we are exploring the design of symmetric-key signatures (i.e., MACs) which operate on short messages and are verifiable by humans with basic numeracy skills.

4.5 On 2-factor Authentication

One limitation of OTP-based systems is that they rely on the use of physical tokens, which are susceptible to loss and theft. For this reason, it is generally recommended that in user authentication tasks, OTPs not be used solitarily, but as a supplement to other tools like passwords or biometrics. As discussed earlier, it is risky to deploy the latter tools in branchless banking systems due to the eavesdropping power available to agents. We make the following suggestion: use numeric passwords (PINs) along with OTPs but leverage human computation capabilities to *combine* OTPs with PINs in a way that foils eavesdropping attacks against the latter. One candidate solution is the substitution-based coding technique of [14], which enables users to mentally transform 4-digit PINs into random 4-digit numbers using 10-digit one-time keys. The transformation scheme of [14] has been shown to be usable by low-literate users and is also a secure 2-factor solution; indeed, Eko, which was involved in developing the solution, plans to deploy it in the near future.

5. CONCLUSION

Despite being celebrated as one of the most promising solutions to financial inclusion in the world, today's branchless banking systems seem to fall short of providing good security guarantees to their beneficiaries. While some may view security of such systems with lesser importance than other design criteria, we believe that security issues are *particularly* a concern here because of the massive cash flows these systems tend to generate and, at the same time, the limited educational background and negotiating power of the users they serve.

This paper begins the process of understanding and resolving the issue of end-to-end security in branchless banking systems. We present the first security model for this problem and are currently working towards designing the first solution which addresses not only the requirements of this model but also the operational constraints under which the solution is expected to be used. Our hope is to arrive at an open set of standards for developing end-to-end secure and deployable branchless banking systems and to improve current practice based on these standards.

6. ACKNOWLEDGEMENTS

Thanks to Bill Thies for reading and commenting on an earlier draft and to the anonymous reviewers of HotMobile 2011. Thanks also to M. Satyanarayanan for shepherding the paper for HotMobile.

7. REFERENCES

[1] A Little World. http://www.alittleworld.com.

[2] Eko India Financial Services Pvt. Ltd. http://www.eko.co.in.

[3] G-Cash. http://site.globe.com.ph/web/gcash.

[4] RSA SecurID. http://www.rsa.com/node.aspx?id=1156.

[5] M-Pesa. http://www.safaricom.co.ke/index.php?id=257, 2007.

[6] SMSSpoofing: Everything you ever wanted to know about SMS spoofing. http://www.smsspoofing.com, 2008.

[7] A. Chaia, A. Dalal, T. Goland, M. J. Gonzalez, J. Morduch, and R. Schiff. Half the world is unbanked. *Financial Access Initiative Framing Note*, Oct. 2009.

[8] R. Chaudhri, G. Borriello, and W. Thies. FoneAstra: Making mobile phones smarter. In *ACM Workshop on Networked Systems for Developing Regions*. ACM, Oct. 2009.

[9] D. Hulton and Steve. Cracking GSM. Talk, Black Hat, 2008.

[10] B. Lorica. Mobile banks in the developing world prove simpler is better. O'Reilly Radar, Sept. 2009.

[11] C. McKay and M. Pickens. Branchless banking 2010: Who's Served? At What Price? What's Next? *CGAP Focus Note*, 66, Sept. 2010.

[12] C. Paget. Personal communication, Aug. 2010.

[13] M. Paik. Stragglers of the herd get eaten: Security concerns for GSM mobile banking applications. In *HotMobile '10*. ACM, Feb. 2010.

[14] S. Panjwani and E. Cutrell. Usably secure, low-cost authentication for mobile banking. In *Symposium on Usable Privacy and Security (SOUPS)*. ACM, July 2010.

[15] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. In *International Conference on Distributed Systems*, pages 509–512. IEEE, Apr. 1981.

[16] A. Sharma, L. Subramanian, and D. Shasha. Secure branchless banking. In *Workshop on Networked Systems for Developing Regions*. ACM, Oct. 2009.

[17] Telco 2.0. Security breach at M-Pesa. http://www.telco2.net/blog/2010/02/security_breach_at_mpesa_telco.html, Feb. 2010.

[18] Telecom Tiger. Nokia forcasts 500 million mobile phone users by 2010. http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html, Apr. 2008.

[19] Wired News. Hacker Spoofs Cell Phone Tower to Intercept Calls. http://www.wired.com/threatlevel/2010/07/intercepting-cell-phone-calls/, July 2010.