Analysis of Two Token-Based Authentication Schemes for Mobile Banking

Saurabh Panjwani, Prasad Naldurg, Raghav Bhaskar
Microsoft Research India
{saurap, prasadn, rbhaskar}@microsoft.com

ABSTRACT

We analyze two token-based authentication schemes, designed for authenticating users in banking systems implemented over mobile networks. The first scheme is currently deployed in India by a mobile banking service provider named Eko with a reach of over 50,000 customers. The second scheme was proposed recently in [1] (in joint effort with Eko) to fix weaknesses in the first one, and is now being considered for deployment. Both systems rely on PINs and printed codebooks (which are unique per user) for authentication.

In this paper, we present a detailed security analysis of the two schemes. We show that Eko's current scheme is susceptible to PIN recovery attacks and a class of impersonation attacks wherein the attacker compromises users' codebooks. The new scheme, on the other hand, is secure against both these attack possibilities. We also show that the two schemes are secure against impersonation attacks where users' codebooks are not compromised. Variants of the new scheme with improved security are also proposed.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection–*authentication.*

General Terms

Algorithms, Security, Human Factors.

Keywords

Mobile banking.

1. INTRODUCTION

In the developing regions of the world, there is a steadily growing interest in using mobile phone networks as a means to extend financial services to the disenfranchised and rural populations. Numerous "mobile banking" facilities have emerged in the last 4 years in countries like India, South Africa, Kenya and the Philippines and today, these systems are responsible for carrying millions of dollars of mobile cash [2,3] to places where conventional bank branches are either completely unavailable or else too inaccessible for the ordinary populace.

Just like in conventional banking, security is a real concern in mobile banking (m-banking) as well, and one of the key aspects of security that needs to be addressed is user authentication: If I, as a bank, receive a message stating "*Move $100 from Alice's account to Bob's account*" over a mobile network, how do I verify that the sender of the message is Alice and not someone else impersonating her? What makes user authentication particularly challenging in m-banking is the fact that a large fraction of mobile phones in the developing world have limited inbuilt security services and are essentially impossible to program with custom solutions. (See figure 1.) Add to this the fact that network-level security on GSM phones has a long record

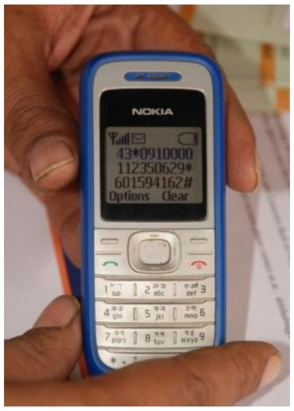


Figure 1. Low-end feature-less phones like these, which neither support GPRS nor are programmable, dominate the mobile landscape in developing countries. It is estimated that over 100 million such phones are in use in India alone [4].

of security vulnerabilities [5] and today's GSM networks neither offer good privacy guarantees to users, nor enable authenticated communication between them [6,7].Together, these factors make the task of designing authentication solutions for mobile banking challenging.

In this paper, we analyze two authentication solutions that have been designed in the context of an m-banking facility named Eko in India [8, 1]. Eko is a business correspondent of State Bank of India (SBI), the leading public sector bank in India, and through its m-banking system, it currently services over 50,000 customers with a daily transaction volume of nearly 2,000,000 Indian Rupees (44,000 USD). The first scheme we analyze in this paper is the one currently being used by Eko across its customer base. The second one is an alternate solution, developed jointly by Eko and Microsoft Research India with the goal to fix certain weaknesses in the former scheme; this scheme is described in [1]. Both schemes rely on the use of numeric passwords (PINs) which are combined with paper-based security tokens – referred to as *codebooks* – to ensure PIN privacy during transmissions (see figures 2 and 3). Implementation of the schemes requires neither any software installation on phones nor any modification of network-layer protocols, which makes them easily deployable on developing-world phone networks.

There is currently no literature on the security analysis of either of these schemes. In [1], although certain weaknesses in Eko's scheme are mentioned and the alternate scheme is claimed to improve security, neither of these claims is substantiated with rigorous arguments[2]. The current paper fills this gap. We first present a detailed threat model that is appropriate for analyzing security of 2-factor authentication schemes in a mobile environment. Our model encompasses PIN recovery attacks and a variety of impersonation attacks that model different amounts of information leakage to the adversary. We primarily focus on security against adversaries who can eavesdrop on users' communication to the bank server and can acquire user's phones and/or their secret tokens. (Both insider and outsider attacks are considered.) Towards the end of the paper, we consider man-in-the-middle threats and outline a technique to these threats in both the schemes; this modification leads to reduced efficiency in the schemes.

The principal outcomes of our security analyses are as follows. First, Eko's current scheme provides poor security against PIN recovery attacks and PINs of users can be completely leaked if the adversary observes just 4 transaction messages, on average. The scheme is also susceptible to an impersonation attack if the adversary acquires the phone and codebook of a user. The new scheme, on the other hand, provides a reasonable amount of security against both these attacks. Second, we find that both schemes are secure against impersonation attacks where the attacker *does not* compromise users' codebooks, although the probability of a successful attack is greater in the new scheme. We then propose some simple variants of the new scheme to improve its security against impersonation attacks.

We remark that although our exposition centers around authentication for m-banking systems, our threat model and the techniques used for analyzing the schemes are potentially applicable in other contexts where user authentication (based on passwords and security tokens) is studied. To the best of our knowledge, the impersonation attack taxonomy developed in this paper is novel and there seems to be no precedent to it in the literature on authentication systems based on passwords and tokens.

2. BACKGROUND AND RELATED WORK

The idea of using mobile phones for conducting banking transactions is not entirely new and has been implemented in developed countries for at least a decade [9]. However, in the developed world, the motivation behind such applications has been to make banking *convenient* for those who already have bank accounts. This paper deals with mobile-based banking systems designed for a different purpose – that of providing *access* to banking for people who do *not* have bank accounts. Throughout this paper, the term m-banking is used to refer to such systems only.

Typical m-banking services rely on a network of human agents who are located close to the target users and who mediate most transactions between the user and the bank. Users approach these agents to enroll into the system and to make deposits into their accounts. A deposit (or a "cash in") transaction requires the agent to send an SMS or a USSD message to a bank server through his/her phone, along with some credentials (like a secret PIN). If the bank server approves of the deposit request, it sends an acknowledgement to both the user and the agent (via SMS/USSD), after which the user must submit the stated amount of cash to the agent, who, in turn, stores it in his cash kitty. Later on, to withdraw cash, this user may approach the same agent or another one and this time, *he* (i.e. the user) would send a transaction message to the bank from his phone, along with suitable authentication information. The bank confirms transaction approval by sending messages to both the agent and the user and the agent then transfers the required amount of cash from his kitty to the user. It has been argued that using agents to mediate banking transactions in this manner is more cost-effective than using regular ATMs in developing regions, owing to the low cost of labor in such places [10], which has led to a lot of excitement and new ventures around the concept. Most m-banking services today also provide a money-transfer facility, using which customers can transfer money from their account into another user's account by sending a suitable transaction message to the bank. Money-transfer transactions typically do not require an agent and thus provide an extremely efficient mechanism to move money across long distances.[4]

There are at least six different services across the world today which are built on this model of banking and over the last 4 years, these services have become a significant contributor to monetary flows in their respective countries. M-PESA, the pioneer of the m-banking concept, today carries over 10% of Kenya's GDP through its network [3], while G-Cash in the Philippines is reportedly transacting hundreds of millions of dollars on a *daily* basis [2]. Given that most users who enroll into such services are motivated by security reasons [10], preventing electronic fraud is of high priority for m-banking services.

[2] The focus of [1] is primarily on a usability comparison of the two schemes and it is shown that the proposed scheme fares better than Eko's in terms of task completion time, error rates and user preferences.

[4] To avoid misuse of such services for, say, money laundering activities, regulatory authorities often place limits on the transaction volumes and balances that customers can maintain.

Current m-banking services primarily rely on PINs – which are normally 4 digits long – to authenticate users to the bank. PIN-based authentication is well-established in conventional banking, and even in m-banking, PINs have been found to be a convenient tool, across a wide spectrum of users [1]. However, for security, it is essential that PINs be protected when transmitted over the network. Different m-banking service providers use different proprietary techniques for protecting PINs but unfortunately, the details of these techniques are often not kept in the public domain. In some situations (e.g., in M-PESA), the service is operated by the network provider, who is in a position to implements proprietary network-level protocols to protect PINs when in transit. To the best of our knowledge, such systems encrypt PINs using GSM's SIM toolkit services; details of the encryption scheme are not publicly known. Other systems like G-Cash seemingly use no encryption at all to protect PINs and there are attacks against such systems already reported [11]. In India, Eko uses a novel interface-layer protocol to protect PINs where PINs are protected through the use of paper-based security tokens.

The use of security tokens for user authentication is a well-established cryptographic technique and several corporate access control systems rely on it, the most popular one being RSA SecurID [12]. Increasingly, banks are becoming interested in deploying such systems as well, particularly for securing Internet-based transactions [13]. Tokens typically contain a list of random one-time passwords (stored electronically or on paper) and each authentication session requires the use a fresh one-time password. Sometimes, randomness is shared across multiple authentication sessions (e.g., the authenticating server sends a set of challenge indices, and the user responds with the random digits or symbols in the token corresponding to those indices). The principal advantage of security tokens is that they provide a factor that supplements the commonly utilized password or PIN to authenticate users. In almost all token-based solutions, the token itself is treated as a secondary authentication factor; the password is given greater importance and maintaining its privacy is regarded paramount. Despite the long history of token-based authentication solutions, research on modeling the security of such systems and on analyzing the security of existing systems is currently lacking; this paper seeks to address this gap in the literature, within the context of m-banking.

Amongst all m-banking systems, Eko seems to be the first to have deployed security tokens as an authentication tool. There have been some proposals to use voice biometrics for authenticating users in m-banking [14,15] but the problem of ambient noise in developing world environments makes such proposals difficult to deploy. Some companies currently use fingerprint biometrics to authenticate users in agent-assisted banking [16,17], but the setup and operational costs of these solutions are significantly greater than that of token-based systems and these solutions are not implementable over low-end mobile phones, which are prevalent in the developing world.

3. THREAT MODEL

We now present a threat model we have developed to define security of token-based authentication solutions implemented for mobile phone networks. The description is kept brief for lack of space; formal definitions of different attack notions will be provided in the future.

We consider adversaries who have complete access to messages sent from users to the bank server and can use this information to mount different types of attacks. Adversaries could either be eavesdroppers on the mobile network (outsiders) who exploit known vulnerabilities of network-layer protocols to recover messages [6,7] or else they could be bank agents (insiders) with whom users interact while conducting withdrawal and deposit transactions. It is reasonable to assume eavesdropping capabilities for agents since in many m-banking systems (including Eko's) agents closely facilitate the communication of withdrawal messages to the bank: the contents of the message, including the authenticating information, are *spoken* out by the user as the agent types them into his or the users' phone and sends them on behalf of the user. Such transactions are often referred to as *aided transactions*. In Eko's current deployment, at least 67% of all withdrawal transactions are conducted in an aided manner, a phenomenon that is attributable to the limited literacy levels of the customers. In such a setting, insider eavesdropping is arguably easier to carry out than outsider eavesdropping.

We consider four different types of attacks against a mobile-based user authentication system. The first is *PIN recovery*, an attack in which the adversary acquires the secret PIN of a user. We then consider three types of *impersonation attacks*, which we refer to as type-0, type-1 and type-2 impersonation attacks. In *type-0 impersonation attacks* the adversary acquires a user's phone and attempts to use it to authenticate to the bank as the phone's legitimate owner. This models a scenario in which a user's phone is stolen or lost and the thief wishes to transact on the user's bank account. In *type-1 impersonation attacks*, the adversary is given, besides the user's phone, access to his unique security token; the goal of the adversary is the same – authenticate to the bank as the legitimate user. Since tokens are susceptible to theft, it is important to guard against type-1 attacks in any protocol design. In *type-2 impersonation attacks*, the adversary acquires not the token but the secret PIN of a user. This models a situation in which a user's PIN gets leaked to a malicious third party.

Security against type-1 and type-2 attacks is necessary to guarantee 2-factor authentication. From a practical perspective, type-0 attacks seem to be most important to prevent against, although all attacks are important to address for ensuring strong authentication. We contend that security against PIN recovery attacks is particularly essential since it is common practice to share PINs across multiple applications and thus, a PIN compromise in a mobile-based application could lead to a compromise in other systems as well. Besides, for the usage of PINs in any authentication system to be meaningful, it is imperative that the system ensure their secrecy, for otherwise, a simpler system which does not use the PIN could accomplish the required task equally well.

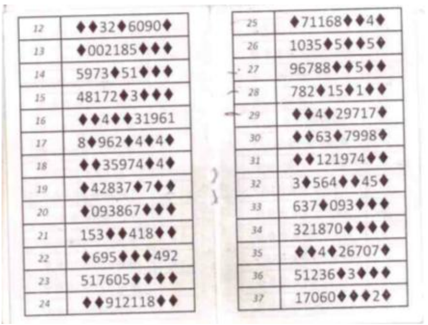
We remark that in all the impersonation attacks we consider, the adversary has access to the phone of the user he is attempting to impersonate. An underlying assumption here is that adversaries cannot easily, and undetectably, connect to the network using arbitrary digital devices and spoof caller IDs of other users. (If that was possible, the requirement of phone possession for impersonation would not be necessary.) Spoofing caller IDs, though shown to be possible in GSM networks [6], requires greater technical sophistication than eavesdropping (particularly when compared to insider eavesdropping), and is thus excluded from our current analysis. Neither of the schemes we consider in this paper provides strong security against spoofing attacks, and for achieving such security, alternate (and conceivably less efficient) techniques seem necessary. The design of such systems for low-end mobile phones, while maintaining usability and simplicity advantages of the schemes considered here, is an interesting open problem, left open by this work.

4. THE SCHEMES

4.1 The Old Scheme

Authentication in Eko's current scheme is based on 2 factors: a secret 4-digit PIN ("what you know") and a codebook ("what you have") illustrated in figure 2. Each entry in the codebook is a string of length 10 and contains a 6-digit random and independently-generated nonce. Interspersed with the nonce are 4 "blank spaces", marked by ♦. The blank spaces are interspersed randomly with the nonce in every string. Both the PIN and the codebook are shared secretly between the user and the bank out-of-band.

In any transaction in which the user needs to authenticate himself to the bank, the user first creates a suitably-formatted transaction message, and appends to that message a 10-digit numeric "signature". Each signature is formed by looking up the first *unused* string in the booklet and by placing the PIN in the 4 blank spaces provided in it. Figure 2 illustrates this with an example. At the other end, the bank server checks if the signature has been formed using the correct PIN and the nonce that is being expected, and only if this is the case, does it process the transaction.



12	♦♦32♦6090♦	25	♦71168♦♦4♦
13	♦002185♦♦♦	26	1035♦5♦♦5♦
14	5973♦51♦♦♦	27	96788♦♦5♦♦
15	48172♦3♦♦♦	28	782♦15♦1♦♦
16	♦♦4♦♦31961	29	♦♦4♦29717♦
17	8♦962♦4♦4♦	30	♦♦63♦7998♦
18	♦♦35974♦4♦	31	♦♦121974♦♦
19	♦42837♦7♦♦	32	3♦564♦♦45♦
20	♦093867♦♦♦	33	637♦093♦♦♦
21	153♦♦418♦♦	34	321870♦♦♦♦
22	♦695♦♦♦492	35	♦♦4♦26707♦
23	517605♦♦♦♦	36	51236♦3♦♦♦
24	♦♦912118♦♦	37	17060♦♦♦2♦

Figure 2. Codebooks used in Eko's current scheme contain sequences of 6-digit nonces, each interspersed with 4 ♦'s that denote blank spaces. For authentication, a user must place his PIN in the blank spaces for the current nonce and thus form a 10-digit numeric "signature". For example, if the user is using the 13th nonce in the codebook (marked ♦002185**♦♦♦), and his PIN is 6391, his signature for the current transaction would be 6**002185**391.**

4.2 The New Scheme

The new scheme proposed in [1] also relies on PINs and codebooks for authenticating users, although the codebooks are constructed differently in this case. Each entry in the codebook is a 10-digit nonce, as shown in figure 4. The digits in the nonce are labeled 0 through 9 to enable users to "look up" digits based on their positions. As before, PINs and codebooks are established out-of-band.

0	1	2	3	4	5	6	7	8	9
5	0	8	1	3	9	2	8	6	7

Figure 4. In the new scheme, each codebook entry is a 10-digit nonce, with the digit positions in the codebook labeled 0 through 9. The scheme involves performing a substation coding of the user PIN using the first unused nonce in the booklet. For example, if the user's PIN is 6391 and the first unused nonce is as shown, the user's signature is the result of looking up the 6th, 3rd, 9th and 1st digits in the nonce, i.e., 2170.

A user with PIN $x_1x_2x_3x_4$ authenticates himself by looking up the first unused nonce in his codebook, and forming a 4-digit number consisting of the x_1th, x_2th, x_3th and x_4th digits in the nonce, in that order. (This is a variant of the well-known one-pad scheme.) This 4-digit number becomes the user's signature for the current transaction. At the other end, the bank re-computes the signature using the locally-stored PIN and codebook.[6] For maximum security, it is recommended that all PIN digits be distinct. There are 5040 such PINs, a space that is sufficiently large to counter dictionary attacks.

[6] In an alternate implementation, the digits in every nonce are forced to be distinct (i.e., nonces are sampled randomly from the set of all permutations of 012..9). Such an implementation facilitates storing PINs in hashed form rather than in plain. (Authentication can be performed by doing a reverse-lookup, followed by hashing the obtained value.) However, the above implementation offers greater security against impersonation attacks.

In our security analysis, we assume that PINs come only from this space.

Nonces in the new scheme are stored in a form so that they can be deleted by the user right after they have been utilized. Several possible storage techniques, like the use of perforated paper sheets, throw-away stickers and electronic hardware, are proposed in [1].

4.3 Synchronization Issues

Synchronizing the user with the bank server is achieved in both schemes is achieved using the same technique. Each nonce is labeled with a unique sequence number (see figure 2) and users must use nonces in order of their sequence numbers. If a user goes off-track, the bank sends an error message with the sequence number of the nonce it is expecting. To prevent dictionary attacks, at most 3 incorrect signatures are tolerated.

There are about 50 nonces in each codebook in each scheme. Users are provided a fresh codebook at the time of registration and every time a booklet gets exhausted or is lost. Upon receipt of a new booklet, users send a codebook registration message (formed using the first nonce in the book) to sync up with the bank.

5. ANALYSIS

The results of our analysis are summarized in table 1. First, we find that the old scheme is insecure against PIN recovery attacks: given a list of *k* 10-digit signatures corresponding to a user, an attacker exhaustively searches for 4-digit subsequences that are common to all of them. If it finds such a subsequence, it reports it as the PIN; else, it aborts. We conducted a small lab experiment with this attack on Eko's scheme. In 3 independent executions on real Eko codebooks, we found that by setting *k* equal to *7*, the attack could *always* recover the PIN. (See figure 3.) On average, across the 3 experiments, every possible PIN could be recovered given just *k=4* signatures.[7] We remark that although this weakness was mentioned in [1], no security analysis with real codebooks was reported therein.

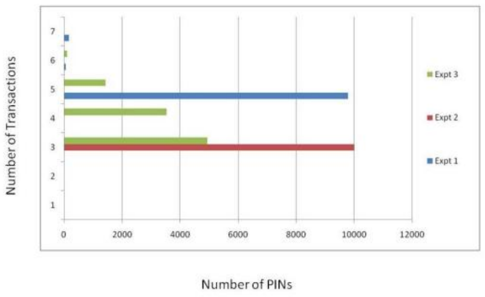


Figure 3: The maximum number of transactions that an adversary needs to observe in order to recover the PIN in Eko's current scheme. The x-axis shows the number of PIN values for which recovery is successful.

On the other hand, we find that the new scheme offers much better security against PIN recovery attacks. The reasoning is simple: assuming perfect randomness of the nonces, the signature computed for any 4-digit PIN with distinct digits is a perfectly random sample from the space of 4-digit numbers. Since the nonces (and thus the signatures) are independent, given any sequence of signatures, every distinct-digit PIN is equally likely to have been used for generating that sequence. The best that any attacker (even an unbounded one) can do is to guess the right PIN and this works with probability at most 1/5040 ~ $10^{\mathbf{-3.7}}$.

Attack name	*Old scheme*	*New Scheme*
PIN recovery	Insecure	Secure ($10^{-3.7}$)
Type-0 impersonation	Secure ($10^{-8.3}$)	Secure (10^{-4})
Type-1 impersonation	Insecure	Secure (~$10^{-3.7}$)
Type-2 impersonation	Secure ($10^{-8.3}$)	Secure (10^{-4})

Table 1: Comparison of the security provisions for the two schemes is shown. The old scheme is completely insecure against PIN recovery and type-1 impersonation attacks. Both schemes are secure against type-0 and type-2 impersonation attacks, though the probability of a successful attack (depicted in brackets) is smaller in the case of the old scheme. Each success probability must be interpreted as being preceded with a multiplicative factor of 3, which is the tolerance level for incorrect authentication attempts.

We find that a similar disparity between the old scheme and the new scheme exists for type-1 impersonation attacks. Since the former is susceptible to PIN recovery attacks, given a user's codebook (and the corresponding phone), an attacker can trivially create signatures on behalf of the user. This does not apply to the new scheme since here, even if a user's codebook (with past nonces having been deleted) and the entire transaction history compromised, from the attacker's perspective, every PIN is equally likely to be the user's PIN. Successful impersonation thus involves guessing the "right" 4-digit sample from a 10-digit nonce, which happens with probability nearly 1/5040 ~ $10^{-3.7}$. (The probability is slightly more than 1/5040 since nonce digits may not be distinct, thus reducing the space of possible signatures.)[8] An attack probability of $10^{-3.7}$ may not be sufficient "in general" but in the presence of a suitable PIN-blocking mechanism – as is recommended for both schemes – it is a reasonable bound. (See the next section for further improvements on security.)

As regards type-0 and type-2 impersonation attacks, both schemes offer reasonable security, although success probabilities are better (smaller) for the old scheme. This is

[7] Part of the weakness lies in the quality of randomness used by Eko to generate nonces. However, even using stronger pseudorandom generators in the lab, we were able to recover PINs from *k=7* signatures with 99.6% success rate.

[8] In the absence of faithful nonce deletion, security against type-1 impersonation attacks is not achieved, since a user's PIN can be recovered using past nonces and signatures.

expected since the authentication information contains more randomness in the old scheme than in the new one. In case of the old scheme, the reasoning is as follows: if an attacker has no knowledge of a user's codebook, then even if he is provided the user's PIN – either directly (type-2) or through a self-mounted PIN-recovery attack (type-0) – he cannot compute the 6-digit nonces for any signature with probability better than random guessing. (Here, we rely on the independence property of the nonces.) Plus, even if he is successful in guessing the right nonce, he must guess the right way to juxtapose the PIN with the nonce to form the 10-digit signature. Assuming that the PIN-nonce interspersion is perfectly random, the success probability of the attacker is $({}^{10}C_4)^{-1} \times 10^{-6}$ which is $10^{-8.3}$.

In the new scheme, without knowledge of a user's codebook, every signature is like a fresh sample from the space of all 4-digit numbers, and this holds true even if the PIN has been compromised. Thus, the probability with which an attacker can successfully create the desired signature for any transaction is at most 10^{-4}.

To sum up, the new scheme offers better security against PIN recovery and type-1 impersonation attacks, which the old scheme fails to counter. On the other hand, both schemes provide reasonable security against type-0 and type-2 impersonation attacks, although success probability of attacks is smaller in the case of the old scheme. Thus, as long as codebooks are not compromised, both schemes offer reasonable security against impersonation, and the old scheme, in fact, offers better security than the new scheme in such circumstances. We next describe ways in which the security strength of the new scheme can be further boosted.

5.1 Improving Security of the New Scheme

The reason for greater probability of successful type-0 and type-2 impersonation in the new scheme is simple: signatures contain fewer random bits than in the old scheme. If an attack resistance of 10^{-4} is deemed insufficient in an authentication application, there is a simple way to improve it to say $10^{-(4+x)}$ for arbitrary x by modifying the scheme as follows: instead of storing 10-digit nonces in the codebooks, store nonces of length $10+x$. Use the first 10 digits of every nonce as before. However, to the 4-digit signature thus obtained, append the last x digits of the nonce as is. Another modification would be to repeat signature generation with multiple independent nonces and to concatenate each of the resulting signatures. A single repetition reduces the success probability of attacks from 10^{-4} to 10^{-8}. Similar techniques can also be applied to the old scheme to improve security against type-0 and type-2 attacks, although these techniques do not improve security against PIN recovery or type-1 attacks in any way.

Note that any such modification of the schemes will affect their usability in the real world. In the context of m-banking, target users have limited educational backgrounds, which raises serious usability concerns and thus makes the above modifications less attractive.

5.2 Tackling Man-in-the-Middle Attacks

One threat that our security model currently does not address is the *man-in-the-middle (MITM) attack*. In such an attack, an adversary can intercept communication from a user to the authentication server and can modify messages while they are in transit. Both schemes we discussed are susceptible to forgeries by a man-in-the-middle (MITM) attacker, as noted in [1]. For example, an MITM adversary can intercept a transaction message with its associated signature, change contents of the message (e.g., the recipient account information) and forward the modified version to the bank. The bank would still view the message as originating from the legitimate user as the signature would be a valid PIN encoding.

While MITM attacks are difficult to mount in mobile networks in real time, we sketch here a solution to counter them in the context of mobile banking. Our solution has some additional requirements: one, all transactions (including money transfers) must be carried out in the presence of a bank agent, and two, the agent must be equipped with a programmable phone. The latter is not an unreasonable assumption to make since the agent phone is shared across multiple users' transactions, and it becomes cost-effective for the bank to invest in such a phone per agent (even where the latter cannot afford one himself).

Once these requirements are met, forgeries can be prevented using standard cryptographic techniques. The agent's phone would be loaded with a public key linked with the bank and appropriate signature verification software. Transaction messages would be sent as before (over USSD) but every acknowledgement from the bank server would be digitally signed using the bank's private key and the signature would be verified by the agent phone. The acknowledgement must contain a unique transaction ID and every piece of transaction-related information that needs to be protected from forgery. Acknowledgements would be sent to both the user's phone and the agent's phone but the verification will occur only on the agent's phone, which is more capable. (Thus, physical proximity to the agent is necessary.) Transactions would be treated as complete only if the signature verification is successful.[9] [10] Designing mobile-based authentication schemes that are secure against MITM-based forgery and that work with

[9] We remark that this solution is still susceptible to phone-rigging attacks by malicious agents who collude with network interceptors. To prevent phone-rigging, physical security must be built into the system using tamper-proof firmware.

[10] By utilizing short signature schemes [19], the information flow from the server to the user can be kept small (an overhead of 20 bytes only) and can even happen via SMS. Plus, the information sent from the user to the bank remains the same as in the original schemes, which is very nominal.

arbitrary low-end phones seems rather non-trivial and is left as an open problem by this work.

6. REFERENCES

[1] S. Panjwani and E. Cutrell, “Secure and Usable Authentication for Mobile Banking,” to appear in *Symposium on Usable Privacy and Security (SOUPS)*, 2010.

[2] B. Lorica, “Mobiles and Money in the Developing World,” *Release 2.0*, Apr. 2009.

[3] High Beam Research, “More Than 10% Of Kenya's GDP Now pass through the M-Pesa Mobile Banking Service,” *http://www.highbeam.com/doc/1G1-193984464.html*, Feb. 2009.

[4] TelecomTiger, “Nokia forecasts 500 million mobile phone users by 2010,” *http://www.telecomtiger.com/fullstory.aspx?storyid=1491*, Apr. 2008.

[5] D. Wagner and I. Goldberg, “GSM Cloning,” *http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html*, Apr. 1998.

[6] D. Hulton and Steve, “Black Hat,” 2008.

[7] O. Dunkelman, N. Keller, and A. Shamir, “A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony,” *Cryptology ePrint Archive: Report 2010/013*, Jan. 2010.

[8] Eko India Financial Services Limited, *http://www.eko.co.in/*.

[9] CNET News, “Bank of America plans to introduce wireless banking,” *http://news.cnet.com/Bank-of-America-plans-to-introduce-wireless-banking/2100-1017_3-228389.html*, Jul. 1999.

[10] G. Ivatury and I. Mas, “The Early Experience with Branchless Banking,” *CGAP Focus Note 46*, Apr. 2008.

[11] M. Paik, “Stragglers of the Herd Get Eaten: Security concerns for GSM Mobile Banking Applications,” *HotMobile 2010: The Eleventh International Workshop on Mobile Computing Systems and Applications*, Maryland: ACM, 2010.

[12] RSA SecureID, *http://www.rsa.com/node.aspx?id=1156*.

[13] Swivel Authentication Solutions, *http://www.swivelsecure.com/*.

[14] Grameen Koota, *http://www.grameenkoota.org*.

[15] A. Sharma, L. Subramanian, and D. Shasha, “ACM SOSP Workshop on Networked Systems for Developing Regions (NSDR),” Montana: ACM, 2009.

[16] A Little World, *http://www.alittleworld.com*.

[17] FINO Limited, *http://www.fino.co.in*.

[18] M-PESA, *http://www.safaricom.co.ke/index.php?id=745*.

[19] D. Boneh, H. Shacham, and B. Lynn, “Short Signatures from the Weil Pairing,” *Journal of Cryptology*, vol. 17, 2004, pp. 297-319.