SMAPs: Short Message Authentication Protocols

Khaled Baqer[1(✉)], Johann Bezuidenhoudt[2], Ross Anderson[1], and Markus Kuhn[1]

[1] Computer Laboratory, University of Cambridge, Cambridge, UK
{khaled.baqer,ross.anderson,markus.kuhn}@cl.cam.ac.uk
[2] Leonine Initiatives (Pty) Ltd., Roodepoort, South Africa
jb@leonine.biz

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

Keywords: Security · Protocols · Usability · Offline · Authentication

1 Introduction

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

J. Anderson et al. (Eds.): Security Protocols XXIV, LNCS 10368, pp. 119–132, 2017.
DOI: 10.1007/978-3-319-62033-6_15

mountains, forests and small islands, they still have to use cash for transactions in their village, and are fully exposed to the depredations of dishonest officials.

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

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

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

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

2 System Model

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

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

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

In some applications, the network may be intermittent rather than absent, and so a side benefit of our protocols is that they can enable payment networks to support delay-tolerant authentication (see Sect. 4.2).

Where both Alice and Bob have smartphones that can communicate directly (Bluetooth, Wi-Fi, NFC, etc.), a very capable offline payment system can be built with conventional tools. However, many poor people use simple GSM phones that cannot communicate with each other directly in the absence of a GSM network. Then the only way for SIM-toolkit applets to communicate across phones is that their users copy sequences of displayed decimal digits. Numerical transactions are already familiar from other systems such as airtime purchases and prepayment utility meters. It is well known, for example, that even illiterate people can cope with twenty-digit sequences provided these are arranged as five groups of four digits, which is enough for a 64-bit ciphertext, and is used in prepayment meters, but is about the usable limit [2].

3 Design Evolution

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

3.1 Basic Protocol

The basic payment protocol proceeds as follows:

1. Alice agrees to pay Bob X and each of them enters both this amount and the other party's phone number into their phones.
2. Bob chooses a 3-digit nonce N_B and forms a 3-digit MAC C (using the shared secret key K) of B and X. He tells Alice the values
$$(N_B, C) \quad \text{where} \quad C = \mathsf{Mac}_K(B, A, X, N_B) \bmod 10^3 \tag{1}$$
with Mac being a 64-bit message-authentication-code function.
3. Alice verifies C. She now believes that she and Bob agree on the amount X and the payer and payee phone numbers A and B (with probability 0.999).

4. Alice authorises the transaction by entering her PIN; her purse decrements its balance by X and generates a 3-digit nonce plus a 4-digit MAC to bind her name and nonce to the data contained in the challenge C:

$$(N_A, R) \quad \text{where} \quad R = \mathsf{Mac}_K(A, N_A, C, N_B, B) \bmod 10^4 \tag{2}$$

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

Verification. We then analysed this protocol via the Burrows–Abadi–Needham (BAN) logic [7]. We idealised the protocol as:

$$\begin{aligned} B \longrightarrow A &: \{B, A, X, N_B\}_K \qquad &(= C) \\ A \longrightarrow B &: \{A, N_A, C, N_B, B\}_K \qquad &(= R) \end{aligned}$$

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

The former follows from our trust in the software, which has the property that it only creates ciphertexts that start with its own identifier. The value X is contained in the challenge C (but see Sect. 3.2). The latter, that $B \mid\equiv A \mid\equiv X$, must be deduced using the nonce verification rule from $\sharp R$ (R is fresh) and $B \mid\equiv A \mid\sim X$ (B believes A uttered X).

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

In our original design, we did not include N_B in Bob's challenge, and as a result could not get the protocol to verify. This is in effect what happens with EMV, which in consequence is vulnerable to the preplay attack [6]. In our initial design, we also wondered whether C should contain Alice's phone number too: $C = \mathsf{Mac}_K(B, A, X, N_B)$. The BAN analysis showed this is superfluous. However we include it for error-detection, as discussed later.

However, despite the fact that the protocol verified, there was an attack.

3.2 Evolution 1: Eliminating Narrow Pipes

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

1. Alice agrees on price X and Bob receives Alice's phone number A.
2. Bob now chooses a higher price X'.
3. Bob then repeatedly feeds X and X' to his SIM_B, which will generate a new nonce N_B each time and output a MAC C. Bob continues until he finds a colliding pair with the same MAC (X, N_B) and (X', N'_B) such that

$$\mathsf{Mac}_K(A, X, N_B, B) \equiv \mathsf{Mac}_K(A, X', N'_B, B) \equiv C \bmod 10^3.$$

This will take just a few dozen trials for a 3-digit MAC.

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

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

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

Such attacks are beyond the scope of BAN, which was only intended for protocols involving cryptographically long nonces and MACs, and does not keep track of guessing entropy or collision risks. Some more modern cryptographic verification tools (e.g. CryptoVerif [5]) can quantify such probabilities.

Anyway, the repaired protocol now runs:

1. Alice agrees to pay Bob X.
2. Bob chooses a 3-digit nonce N_B, forms a 3-digit MAC C with B and X, and sends Alice:

$$(N_B, C) \quad \text{where} \quad C = \mathsf{Mac}_K(B, A, X, N_B) \bmod 10^3 \tag{3}$$

3. Alice verifies C to ensure they agree on payer, payee and amount.
4. Alice authorises the transaction by entering her PIN; her purse decrements its balance by X and generates a 3-digit nonce plus a 4-digit MAC:

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

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

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

3.3 Evolution 2: Transaction Chaining

The revised protocol still raised security concerns:

1. Bob could try to add money to his SIM card by faking transactions with fake customers and just guessing the response R. Each attempt will succeed only with probability 10^{-4}, and repeated attempts can be blocked by a retry counter. We can also mix up R with N_A. However the detailed design is far from trivial. For example, Bob could connect SIM_B to a PC to automate an attack, and interleave guessing attempts with real payments from a customer SIM he controls. In the worst-case attack, Bob is the local payment service agent, and has hundreds of customer SIM cards in stock. We need to make the fraud probability acceptably low without making the authorisation code too long or otherwise making operations too complex or fragile.

2. Alice can similarly fake a transaction with probability 10^{-4}, but this may be less of a concern in practice, as Alice cannot repeat thousands of transaction attempts against SIM_B without the collaboration of Bob. But a colluding Bob could just run the attack without Alice (see above).
3. Bob can also try to fake transactions with real customers A, by keeping a record of their $\mathsf{Mac}_K(A, N_A, X, N_B, B)$ replies. In such a fake transaction, Bob can choose A and N_A, and if the real Alice has already paid n times in the past for a regularly bought item of fixed price X, then Bob has enough data to be able to look up a valid pair (N_B, R) to complete a fake transaction with probability $n \cdot 10^{-3}$.

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

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

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

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

$$T_0 := (A, B) \tag{5}$$

$$S_0 := H(0, T_0) \tag{6}$$

$$i := 0 \tag{7}$$

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

SIM_A and SIM_B also set up a shared transaction key K_{AB}. At this stage we can assume it is simply derived from their phone numbers using a key derivation function and the universal shared secret K: $K_{AB} = h_K(A, B)$. (In Sect. 4 we discuss options for mitigating the risk of a shared master secret.)

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

1. Alice and Bob agree on a price X.
2. Alice and Bob then select the payment session to be used, and their SIMs retrieve not just A and B but also the hash chain values (i, S_i).
3. SIM_A checks that X is within limits agreed with Sam, and that Alice's on-card purse value M_A has enough funds for the transaction, presumably $M_A - X \geq 0$. It aborts the transaction otherwise. Likewise, SIM_B checks whether both X and its purse value $M_B + X$ are within limits agreed with Sam, and aborts if not. (Transaction limits agreed with Sam may be variable, and depend on for example X, i, A, or B.)
4. Bob's SIM_B generates and displays a challenge of $n_{\mathrm{c},1}$ deterministic digits that Alice and Sam can verify, $n_{\mathrm{c},2}$ deterministic digits that Sam can verify, as well as $n_{\mathrm{c},3}$ digits of random entropy to help ensure that payment sessions do not repeat – an $n_{\mathrm{c}} = (n_{\mathrm{c},1} + n_{\mathrm{c},2} + n_{\mathrm{c},3})$-digit decimal challenge message

$$\begin{aligned} C_{i+1} = E^{n_{\mathrm{c}}}_{h_{K_{AB}}(S_i)}[(\mathsf{Mac}_{K_{AB}}(i, S_i, X) \bmod 10^{n_{\mathrm{c},1}}) \parallel \\ (\mathsf{Mac}_{K_B}(i, S_i, X, F_B) \bmod 10^{n_{\mathrm{c},2}}) \parallel N_B] \end{aligned} \tag{8}$$

which Alice copies into her phone. Here $0 \leq N_B < 10^{n_{\mathrm{c},3}}$ is a number picked by SIM_B and $\parallel$ is concatenation of decimal digits. $E^{n_{\mathrm{c}}}$ is a pseudo-random permutation over $\mathbb{Z}_{10^{n_{\mathrm{c}}}}$ (see [4]), to ensure that adversaries cannot be certain about the function of individual digits without knowing K_{AB}. F_B is optional other information that can be included and that SIM_B will eventually transmit to Sam (but not to SIM_A) such as M_B or a timestamp, added as an entropy source and as forensic evidence of disputed transactions.
5. Alice's SIM_A then calculates an $n_{\mathrm{r}} = (n_{\mathrm{r},1} + n_{\mathrm{r},2} + n_{\mathrm{r},3})$-digit response

$$\begin{aligned} R_{i+1} = E^{n_{\mathrm{r}}}_{h_{K_{AB}}(S_i, C_{i+1})}[(\mathsf{Mac}_{K_{AB}}(i, S_i, X, C_{i+1}) \bmod 10^{n_{\mathrm{r},1}}) \parallel \\ (\mathsf{Mac}_{K_A}(i, S_i, X, C_{i+1}, F_A) \bmod 10^{n_{\mathrm{r},2}}) \parallel N_A] \end{aligned} \tag{9}$$

containing $n_{\mathrm{r},1}$ digits that Bob can verify, $n_{\mathrm{r},2}$ digits that Sam can verify, and $n_{\mathrm{r},3}$ digits N_A chosen by SIM_A. F_A is optional other information that SIM_A will record and eventually transmit to Sam, such as M_A or a timestamp.
6. SIM_A then updates its non-volatile state and on-card transaction records:

$$M_A := M_A - X \tag{10}$$
$$i := i + 1 \tag{11}$$
$$T_i := (S_{i-1}, X, C_i, R_i) \tag{12}$$
$$S_i := H(i, T_i) \tag{13}$$

7. SIM_A finally displays its response message (9).
8. Bob types R_{i+1} into his phone, which reports the success or otherwise of the transaction.

The point of hash chaining is twofold:

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

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

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

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

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

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

4 Mitigating the Risk of a Shared Master Secret

4.1 Evolution 3: Group Key Scheme

If smartcards were perfectly tamper-resistant, then the initial design would be largely complete at this point. They are certainly more resistant than was the case twenty years ago, when large-scale compromises of pay-TV smartcards were pretty well an annual occurrence. This is partly due to technological improvements such as side-channel countermeasures, top-layer sensor meshes, and randomised place-and-route; and partly due to minimising the value that can be extracted by cloning a single card. In EMV, for example, the keys authenticate transactions on a single account, so the extractable value is typically in the low tens of thousands of dollars, rather than the millions that could be made from cloning a pay-TV card. As long as reverse engineering a card costs tens of thousands of dollars and the attacker needs perhaps a dozen identical cards to work with, this is probably enough.

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

The downside with this approach is that a substantial volume of LDC phone payments are migrant remittances, where neither the sender nor the receiver is a merchant.

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

A similar idea can be adapted here, where the communications are many-to-many. For example, we can divide all users into $d = 100$ key groups, give everyone 100 out of 5,050 keys, and thus require an attacker who wanted to impersonate any user to clone 100 cards rather than just one.

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

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

$$g_A := h_{K_{\mathrm{g}}}(A) \bmod d \tag{14}$$
$$g_B := h_{K_{\mathrm{g}}}(B) \bmod d \tag{15}$$

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

$$G_A = \{K_{0,g_A}, K_{1,g_A}, \ldots, K_{g_A-1,g_A}, K_{g_A,g_A}, K_{g_A,g_A+1}, \ldots K_{g_A,d-1}\}. \tag{16}$$

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

SIM$_A$ and SIM$_B$ can now secure their message exchanges using the key

$$K_{AB} = \begin{cases} h_{K_{g_A,g_B}}(A,B) & \text{if } g_A \le g_B \\ h_{K_{g_B,g_A}}(A,B) & \text{if } g_A > g_B. \end{cases} \tag{17}$$

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

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

4.2 Evolution 4: Delay-Tolerant Needham–Schroeder

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

An initial idea is to turn the bug in the Needham-Schroeder protocol [8] into a feature. The fact that Bob has no guarantee of the freshness in Alice's initial exchange with Sam, and that she can therefore pass on a key packet to him a month or even a year afterwards, can be used to create a delay-tolerant version of the protocol. In order to preserve the original notation of [8], we reverse the roles in this iteration whereby Alice (A) is the merchant travelling to the city (data mule), and Bob (B) is the housebound villager who requires relayed messages from Sam (S). We assume that Alice and Bob have just done the first transaction in a new payment session, and want Sam's help to set up a K_{AB} that is present only in their SIM cards and thus cannot be compromised if any other card is reverse-engineered.

1. Alice catches the boat into town, and SIM$_A$ detects a connection. SIM$_A$ automatically uploads transaction logs to Sam (to update shadow accounts for reconciliation). She also sends a request to authenticate with any new counterparty, in this case Bob[2]:
   $A \rightarrow S : A, B, N_A$
2. Sam sends K_{AB}, via SMS or USSD, encrypted with the shared key K_{AS}:
   $S \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. When Alice returns to the village, SIM$_A$ can display, during transactions discussed below, the authenticated shared key:
   $A \rightarrow B : \{K_{AB}, A\}_{K_{BS}}$.
4. The final two steps of the Needham–Schroeder protocol are performed offline between the transacting parties, by showing authenticated nonces to obtain agreement:
   $B \rightarrow A : \{N_B\}_{K_{AB}}$
   $A \rightarrow B : \{N_B - 1\}_{K_{AB}}$

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

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

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

[2] The notation here follows the original Needham–Schroeder protocol [8].

5 Usability

Our goal was to achieve the most usable design that is acceptably secure. We present here a cognitive walkthrough and error recovery analysis.

First, both customer and merchant have to enter each others' phone numbers. This happens anyway in phone payment systems, and there is a mistake every few attempts. Even trained people who are alert make about one digit-entry error per 200 digits [3], and so we expect at least one in ten attempts to sync phone numbers to be problematic. In existing systems, error recovery can waste a call or a text. In our proposed system, an error is detected when Alice verifies Bob's challenge.

After Alice has entered Bob's six-digit string and got an "OK", she enters her PIN to authorise the payment, just as at present. She then shows, or calls out, the seven-digit payment code to Bob. If it's accepted by his SIM card, his phone will show "OK". If she reads it wrong, or he types it wrong, then neither has the money: her value counter has been decremented while his hasn't been increased yet. With good faith, they both have an incentive to retry until they get it right; Alice wants the goods, after all, and Bob wants her money. Even if Bob's phone battery suddenly goes flat, they can write down the seven digits and complete the transaction later.

If they are interrupted or never complete, then when Bob and Alice later upload their transactions, Sam will have the data needed for dispute resolution.

If Bob is trying to cheat Alice by getting her to pay twice, he can report an "OK" as a failure. Alice will detect the cheating on reconciliation. There is however an issue here about whether Bob might abuse a position of power, or Alice's illiteracy. Similar attacks are possible on many other payment systems (e.g. pension payments) and are mitigated by social rituals. We will experiment with poster instructions that Bob should show Alice the results of typing in the payment code; if this is overt so that everyone in the shop can see it, the exploitation risk should be acceptable. Another possibility is a close-out code from Bob, which we plan to field test.

6 Conclusion

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

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

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

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

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

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

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

References

1. Anderson, R.J.: UEPS – a second generation electronic wallet. In: Deswarte, Y., Eizenberg, G., Quisquater, J.-J. (eds.) ESORICS 1992. LNCS, vol. 648, pp. 411–418. Springer, Heidelberg (1992). doi:10.1007/BFb0013910
2. Anderson, R.J., Bezuidenhout, S.J.: Cryptographic credit control in pre-payment metering systems. In: Security and Privacy, p. 15. IEEE (1995)
3. Baddeley, A., Longman, D.: The influence of length and frequency of training session on the rate of learning to type. Ergonomics **21**(8), 627–635 (1978)
4. Black, J., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (2002). doi:10.1007/3-540-45760-7_9

5. Blanchet, B.: CryptoVerif: computationally sound mechanized prover for cryptographic protocols. In: Dagstuhl seminar Formal Protocol Verification Applied, p. 117 (2007)
6. Bond, M., Choudary, O., Murdoch, S.J., Skorobogatov, S., Anderson, R.: Chip and skim: cloning EMV cards with the pre-play attack. In: IEEE Symposium on Security and Privacy (SP), pp. 49–64. IEEE (2014)
7. Burrows, M., Abadi, M., Needham, R.M.: A logic of authentication. Proc. Roy. Soc. Lond. A Math. Phys. Eng. Sci. **426**, 233–271 (1989). The Royal Society
8. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM **21**(12), 993–999 (1978)