SMAPs: Short Message Authentication Protocols (Transcript of Discussion)

Khaled Baqer and Ross Anderson

Computer Laboratory, University of Cambridge
{khaled.baqer,ross.anderson}@cl.cam.ac.uk

Khaled Baqer: What I'd like to do first is to highlight the background and motivation for the payment project that we're working on at Cambridge. I'll do a ten-minute introduction and Ross will take over to discuss some of the attacks and the interesting parts of this paper.

The story begins with the mobile payment revolution. This is not Apple Pay, Google Pay or whatever extension of EMV you have on your phone. These are mobile payments in LDCs, less developed countries. So top right, you have M-Pesa in Kenya, top left you have bKash in Bangladesh. These systems work by providing a menu for the user on the phone's SIM toolkit (applet running on the SIM), so they can access, for example, 'send money', enter the amount and the recipient's phone number. Usually this is saved in the user's phonebook, so all you have to do is to enter the amount. The phone sends a message to the server, and the server replies with a confirmation back to the user, and the recipient gets a confirmation of the transaction.

This has been transformative in bridging the gap and providing financial inclusion. Millions of people don't have bank accounts, and even if they do, these accounts are dormant and nobody really uses them. Because of social reasons we can discuss offline, people don't want to go to banks. Some of them don't even know how to use their accounts, and they don't want to walk miles to the closest bank branch.

The first big use for M-Pesa was remittances: someone working in Nairobi sending money back home to their relatives in a remote village, and then they can cash out with the local agent. Store-of-value, of course, and personal safety, these are big ones: you don't want to carry cash in some regions, it's unsafe. It also provides a means for governments to deliver payments directly to users. Meaning if they want to have a direct financial relationship with the users they can directly send the funds to their phones and not send cash that can be lost along multiple hops, if it gets delivered to the intended recipient at all.

What we started out with are the challenges listed here: can we extend this to areas where there is no network? Because mobile payments work beautifully if they work, but they stop when the network does, and so we have people in some places walking miles to the closest area where they can get reception on their phone, just to make a phone call.

This is not only limited to LDCs. There was the blackout in the US years ago, and more recently, in the UK you have regions that were offline because of a power outage and people could not use their cards, they could not use mobile devices to pay, they were running out of cash, and it was getting horrible.

Another reason we want to do this is to cut network charges. We think that transaction fees are a big hindrance to increasing the uptake of mobile payments in less developed countries. When the transaction fee is a big chunk of the transaction itself, people will just use cash.

The main constraint is that we have to design this for basic phones. They're called feature phones, but really they lack any features: no cameras, no NFC, no Bluetooth, absolutely nothing. These are £20 or less no-camera phones.

I'm sure you can come up with more examples than the ones shown on the slides about short message authentication protocols, but they're familiar from the three-digit CVV codes that you can find on the back of your bank card. Of course, it's familiar to the audience of this workshop that usability and security go hand in hand. We're limited with regards to what security mechanism we can provide, based on the usability of the system, and if the users see that there's a point of using something like this. I talked about the offline constraint environment, and now we will discuss all the problems in the context of offline payment systems in less developed countries.

We have existing offline purse systems. We have Geldkarte in Germany, we have UEPS, if you're familiar with Ross's work. These systems can be implemented in SIM toolkits, which is what we want to do. The first problem, then, is how do we access the SIM toolkit, because feature phones have one SIM slot and that's already taken by the mobile network operator (MNO), and we don't have access to that.

This is the initial barrier for entry, because we can't access a trusted zone in the user's phone. Another problem is that these protocols are designed for complex messages back and forth between the devices and coupled with the third problem, that the phones don't have any bandwidth for communication, this becomes something that we cannot implement while maintaining some sort of reasonable usability.

So, how do we solve the first problem? We now have the enabling technology to bypass the MNO's restriction on the device. This is a SIM overlay, that Ross, I think, has one of them, if you care to see it. It's a trusted element, it acts as a SIM and looks like a very thin SIM. It's a sticker. You can peel it off, you can put it on the existing SIM (the MNO's SIM), then insert it back into the phone. This gives the user access to two SIM toolkit menus.

Now we have a trusted zone that we can program. It's a Java card, and it's very simple to program. We can load our own keys, we can do whatever we want in this trusted element. That's the first problem solved.

What we want to present here is DigiTally, the offline payment system that we're working on. We want this to be free and open-source to get the largest adoption possible; no patent issues with the systems that I mentioned before.

We want this to be implemented as SIM toolkit applets. Smartphone apps would make life a lot easier, but we want something deployable today, not in five to ten years when Android phones become more popular than they are right now. So we want to implement them in overlay SIMs, as a proof-of-concept to demonstrate that an MNO's restriction is not an obstacle.

The name, DigiTally, and I credit Ross for that, is due to the fact that the system works by means of tallying the digits that are exchanged between the users. So how does this work? I will now describe the basic protocol. You can probably see a lot of problems here that Ross will definitely summarise for you, but this is the strawman stuff.

Alice and Bob want to pay each other X. They know their identities, A and B, and they agree on X. What Bob the merchant does first is to generate a nonce N_B. He MACs the transaction and gives Alice N_B, B (if unknown to Alice) and C which is a subset of the MAC (a few digits), not the entire MAC. Alice receives this verbally from Bob, or the code is shown to Alice. She enters the digits in her phone, and if the MAC verifies on both devices, she authorises the transaction with her PIN. Her device does something similar: she generates another nonce N_A, and she generates the response R to Bob's challenge C.

In this response, she includes the two nonces, N_A and N_B, she includes the challenge MAC that she received, and she includes the two identities A and B and she MACs all these parameters producing R. She provides N_A and a subset of R to Bob. Bob enters the digits into his phone, if everything checks out, then Alice's balance was already decremented by X, and Bob's device is incremented by X (after entering R). I will let Ross go through the interesting attacks and development of the protocol.

Ross Anderson: Well, so we've got a protocol that appears to work. This being the protocols workshop, one of the first things that you do is try to verify it. So can you verify it, for example, using the BAN logic? Summarising the protocol here, the challenge is a MAC of Bob's phone number, Alice's phone number, the amount and Bob's nonce; and the response is a MAC of Alice's phone number, Alice's nonce, the challenge that was just seen, Bob's nonce, and his phone number.

The reason for the challenge is that you want to make sure that both of the parties agree on the sender phone number, the recipient phone number and the amount. The reason for the payment nonce is, of course, to show that the payment was authorised. Now this can be built into the standard transaction flow for M-Pesa and the other mobile phone systems with the addition of the challenge and response. I should perhaps mention that most of these mobile phone payment systems use essentially the same transaction flow, because they started off using software from a firm called Fundamo that ended up being bought by VISA. So we take the standard system, and we put a small modification on it: the challenge and response.

What we need is that Bob trusts X, if Bob believes, Alice believes X, and if Bob believes that Alice has jurisdiction over X. Why should Alice believe X? Well, if Alice uttered X and X is fresh, according to the nonce verification rule, and all of these trace back to the software constraints, so this appears to be fairly straightforward.

Jonathan Anderson: Where does K come from?

Ross Anderson: K is the key that they share, and I'll discuss that later. To begin with, you can think of it as being a universal shared master secret, and we'll refine that concept as we go on. In fact, our version 0 design didn't verify, because we didn't put N_B (Bob's nonce) in R, and this meant that it failed to verify. Then we realised this is exactly the same vulnerability that you have in EMV, which leads to the pre-play attack. The use of the BAN logic in this case enabled us to avoid a common error from which millions, billions of bank cards in the west suffer, and which leads to fraud. Are we all sorted then? Have we got a verified protocol?

Bill's shaking his head, and of course that's absolutely right.

Bill Roscoe: I've read the paper.

Ross Anderson: He read the paper. He cheated. That's cheating! [laughter] Can anybody else see what the vulnerability is? It turns out that the vulnerability is the challenge C is only three digits long, and so you can do, basically, a birthday attack on that, which is outside the scope of the BAN logic. The BAN logic assumes that all nonces, all keys, all MACs and so on, are infinitely long. It doesn't take account of entropy, and in fact that's the case with most verification tools with the exception of CryptoVerif. How do you do an attack? Well, Bob chooses a higher priced X. Say X is 500 Kenyan shillings for a bag of rice and X might be 50,000 Kenyan shillings, and so Bob generates a series of new nonces NB and he finds a collision such at the MAC of A, X, N_B and B is the same as MAC of A, X, N_B and B. As C is only three digits, that means that you have to make several dozen attempts, which is quite feasible – even on a phone with a rubber keyboard.

Bob then aborts all of the trial transactions except for the last one of the colliding pair, and he can give N_B and C to Alice, but in his SIM he uses N_B and X. This means that when the transaction goes through, Alice pays 500 Kenyan shillings, and her value counter is decremented by that much, while Bob receives 50,000 Kenyan shillings and his value counter is incremented by that much. This violates the law of conservation of money, on which banks are understandably rather keen. So this is a significant failure.

This is one of the interesting things from the scientific point of view that comes out of considering short message authentication protocols. You can't just use your raw, out-of-the-box verification tool kit, because you have to start keeping track of entropy as well.

Incidentally, it was our colleague Markus Kuhn who spotted this 'man in the middle' attack, which is how he joined the paper as an author.

Then we got together and we started thinking about other issues around entropy. And the obvious thing is that Bob could try to add money to his SIM card by faking customers and just guessing the response R. If you're guessing a four-digit response, then that means that you might have to make 5,000 guesses.

Are you worried about that? How hard are people prepared to work, in a country where people earn a dollar a day, in order to make some upper transaction limit? And how feasible is it for you to implement on your SIM card, velocity checks as to how many bad MACs can be verified before you have to go online and refresh the system? That's a bigger design issue that we discuss in the paper.

What we can also do is look for collisions among transactions with real customers. Is it possible to confuse a former transaction X with former transaction Y? Now there's a generic fix to these kind of attacks, which is that you generate all the nonces with a key K_{AS}, known to Sam the banker. Alice's nonce N_A is some counter encrypted under K_{AS}. K_{AS} is present in Alice's overlay SIM card and also in Sam's hardware security module in Nairobi. This way, when transactions are uploaded, Sam can check the nonces as well as the MACs, and if something is wrong then he can alarm on that.

That is basically a variant of trick that was already used in UEPS 20 years ago, in that of the things that get uploaded, you have some things that could be checked by the merchant and other things that could be checked only by the bank. This is one of the ways of dealing with the risks of such systems.

The second thing that we start to think about as a second evolution, is why don't we chain transactions together? We started thinking about this, because typical mobile phone payment transactions in the third world very often go to people to whom you have paid before. Either you are paying the village storekeeper again and again and again, buying shopping bags full of household necessities; or else another big application of such systems is remittances. You go and work in Nairobi, and you send money back to your wife or to your mum, or whoever, in the west country once a week.

If you can establish a long-lived session between payer and payee, then perhaps in the case of small repeated transactions you could use fewer MAC digits, say when you're making your hundredth purchase from the village store for 500 shillings. You could then see to it that you can block various attacks which involve guessing or finding collisions with past MACs. What you can do is replace N_A and N_B with MACs of the transaction data, and you can keep shared state between Alice and Bob, which means that if somebody gets a fortuitous collision on a MAC, it will only work for one transaction.

The sort of thing that you do is have the payment session being a hash chain, and the status is a hash digest and a counter. When Alice and Bob start a new transaction, the SIMs initialise a session, and then whenever you do another transaction, you update the state. I'm not going to read through this equation. You can get the details in the paper.

This means that you can tune the transaction parameters for the number of MAC digits you want on initialisation, the number of MAC digits you want at each transaction, and the risk of fraud. How you go about verifying protocols like this, we don't really know. We suspect that you have to write down some kind of game and use game theory to look for equilibria – circumstances under which some particular fraud strategy might give a payoff. That we leave for future work. And again, that's the second part of the transaction chaining. Then what you do is update the state and decrement the value counter, and the transaction proceeds as before.

The third evolution that we considered is: what about group keying? If overlay SIMs had enormous storage, you could simply give every SIM card a unique K_{AB} for every other SIM card. Right? Then you would have unique pair of keys and bank rules generally say something like that for an EAL4 verified card you can do transactions up to 20 Euros or 20 dollars, or whatever. With a universal shared secret key, but for larger transactions you're expected to have a unique key per transacting pair of Alice and Bob.

Is there a useful halfway house? Well, broadcast encryption schemes from about 20 years ago had the idea that you could prevent the break of a single pay-TV card from being used to generate a universal solution to a satellite TV system simply by dividing your subscribers into groups. The same idea happens here. What you can do is use combinatorial keying whereby you divide users into a hundred key groups, and you then give each SIM card a hundred keys out of a total ... sorry, that's a mistake. That should be 4,950 keys: half n, n minus one. That means that an attacker who breaks one card could then only impersonate one percent of the card fleet. He can't impersonate a hundred percent of the card fleet. The idea here is that you can push out the costs of somebody doing a group break.

Why do we care about this? Fraud happens when it's industrialisable, and how is somebody going to industrialise an attack on a system like this? Do you extract value from a small number of merchants who happen to have a lot of cash on the premises, or do you attack the system by handing out SIM cards that enable lots of people to go and take small amounts from their village merchant? One works with these case studies and comes to the conclusion that in many of these cases, a class break would involve drilling keys out of more than one card if you use group keying.

This isn't, however, a magic solution, because this is easy to do in the case where the bank is the phone company, but a system like this is more likely to be taken off in cases where the bank isn't the phone company, because then the bank is having to pay the phone company for SMSs, and has got an incentive to deploy a system like this.

Now here's the really interesting bit: Delay-tolerant networks. There's a lot of places in the world where people have to deal with intermittent network service. One case: we came across a story of a village in the Finmark, in northern Sweden, where there isn't GSM service and there is a helicopter service in the nearest town, so what people do is they get their phones and give them to the helicopter driver, and rather than paying for a helicopter ride yourself, which is several hundred Euros, you just get your phone taken into town and it gets your email updates, your Facebook updates and all the rest of it. There are similar things apparently done in villages in Brazil, where you have to get on a speedboat for an hour or two into Manaus, in order to check your Facebook.

Wouldn't it be neat if you could have a PC in the village which would enable local sharing, and which could also be upgraded by means of a USB card carried every day by the speedboat driver? Getting the Internet and getting online service to the last billion people is probably going to involve thinking about the infrastructure for delay-tolerant networks.

So is there any general mechanism that might be applicable here, and that we could use in our case of off-network payments in Africa and South Asia? Well, the case that's perhaps of most interest is where people want to do relatively high value payments, again and again, to the same recipient. Again, think the guy who goes into Nairobi and gets a job, and wants to send money home to a remote village once a week, over the ten pound limit. So what you want to do is have a means whereby after some initial protocol or handshake, Alice and Bob end up with a high-quality, long-term shared key which they can use to authenticate transactions above that universal key limit.

Remember Needham-Schroeder? The shared-key version? We all tell our students that this was one of the first crypto protocols, but it's got a bug. Right? Alice says to Sam, I want to speak to Bob and here's my nonce. Sam says here's a key encrypted for you, and here's the same key encrypted for Bob. Alice goes to Bob and says, here's a key packet from Sam which enables you to speak to me. And of course the bug is that Alice could wait for a year between the second message and the third message, and so you've got no guarantee of freshness, at least from Bob's point of view. And this is considered to be a bug that was fixed in Kerberos by moving to timestamps.

But we realised, in the delay-tolerant network environment, this isn't a bug. This is a feature. A well-studied protocol, well out of patent, does exactly what we want! Because Alice and Bob want Sam's help to establish a unique shared key, K_{AB}, so that they can then do transactions above the twenty dollar limit. One party starts a Needham-Schroeder protocol with Sam once they get connectivity, okay? Alice and Bob do their first ten dollar transaction, or whatever, using the shared key that's in every SIM card, but then both of the SIM cards remember, and Alice's SIM card says, 'I remember I've done a transaction with Bob. Opportunistically, I should get a K_{AB} whenever I can next speak to Sam.'

Alice then goes online, and you get the key packet. The thing that you then have to work on is how do you exchange the digits offline for the key packet K_{AB} encrypted by K_{BS} – that's the third message there – with the other party? If you're going to encrypt something in a single AES block, but that's 128 bits, which is 40 decimal digits, which is perhaps a bit of a pain. How many digits do you actually need for K_{AB}? How many bits do you need for K_{AB}? Will 80 bits be enough? Are 64 bits enough? How big does the key packet have to be?

Well, one of the things that we do know is that from the work in pre-payment electricity meters we did 20 years ago, that people – even people who can't read and write – can easily manage 20-digit codes, because 20-digit codes presented as five blocks of four digits are how people buy electricity in many less developed countries. That we know we can make work. Is that good enough? Can people deal with 30? Can people deal with 40? That's for experimentation.

The next step in payment networks, we believe, is to extend them to areas where the network stops, and what we hope to do this year is a realistic field trial so we can figure out whether people can use this stuff, and what the usable parameters are for challenges, for responses, for key packets, for delay-tolerant

Needham-Schroeder, and the constraint here is the user interface. That's what we're designing out from. The crypto, we assume we can do. Okay, we may need two or three goes and great care over the verification, but what we've got to do is have something that people will actually use in the field. We need to think about scalability and recovery and so on, if it's going to be acceptable to banks.

What we've shown in the paper, which I hope you'll read, is how we evolved the protocols to deal with entropy problems which the BAN logic missed, and which other verification techniques mostly miss – dealing with transaction sharing, dealing with mitigating the risk of shared keys. And the future research direction is what do we do with delay-tolerant networks in general, because these are going to be important. They were important in the past, when a lot of stuff was offline. We've tended to forget them over the past 20 years, but we're now seeing more and more stuff require some delay-tolerant capability – EMV for example. How much more is going to be needed and how do we as protocol engineers go about delivering on that requirement?

That's what we've been up to, and that's what we're planning to do in field trials during the summer. Any questions?

Vashek Matyas: How low-level is the mobile that you are considering in this scheme?

Ross Anderson: The mobiles that we're considering are the mobiles that you buy for $8.00 in Tesco's. They have no camera, they have no touch screen, they've got rubber keyboards, and they've just got a little LCD display with several lines of ASCII text. That's what most people have in the demographic that we're trying to reach. You see, the reason we did this is that the Gates Foundation had a call for proposals for how to do payments off-network, and so we've come up with this idea and won a small grant which should pay for the field trials.

Vashek Matyas: But with a camera you can rely on bar or QR codes?

Ross Anderson: With anything like that, you're easy. If you've got NFC or Bluetooth or anything like that, you just do all this in an app. But given that we've got the mechanism, you can use the same mechanism whether it's in an app or whether it's being done by manual copying of digits.

Jeff Yan: You actually answered my question already. I was curious on what motivated the design and how did you get realistic requirements.

Ross Anderson: The Gates Foundation has funded dozens and dozens of mobile payment networks in LDCs. They found it transformative for growth and development, but it runs out where the network does, and there's lots of villages that are off-network.