Make noise and whisper: a solution to relay attacks

Omar Choudary and Frank Stajano

University of Cambridge Computer Laboratory
15 JJ Thomson Avenue, Cambridge, CB3 0FD, United Kingdom
first_name.last_name@cl.cam.ac.uk

Abstract. In this paper we propose a new method to detect relay attacks. The relay attacks are possible in many communication systems, and are easy to put in practice since the attackers don't require any knowledge about the underlying protocols or the cryptographic keys.

So far the most practical solutions against relay attacks rely on distance-bounding protocols. These protocols can provide an estimated maximum distance between two communicating devices.

We provide a different solution that can detect a relay attack regardless of the distance between the devices. Our solution relies on introducing intentional errors in the communication, providing a kind of hop-count metric.

In order to illustrate our idea we describe two idealized example implementations based on bidirectional communication channels. Based on simulation experiments we assess the effectiveness of these methods. There are several limitations in these two examples but we hope that the ideas presented in this paper will contribute towards practical implementations against relay attacks.

1 Introduction

Many systems are vulnerable to a relay attack, where an adversary can forward data, even encrypted, between two communicating parties in order to obtain some benefits (e.g. money or access to a building). Several solutions have been proposed to solve this problem, of which we mention distance bounding and multichannel protocols. These methods require precise time measurement or extra communication channels, respectively.

We offer a new approach to detect relay attacks that does not rely on distance bounding and which is distinct from multichannel protocols (though vaguely related). Our approach relies on creating a kind of hop-count metric based on introducing noise in the channel.

We suggest two possible physical implementations of the approach and discuss their advantages and limitations.

2 The relay attack problem

A relay attack can be imagined as the extension (or even conception) of the communication channel between Alice and Bob without their knowledge. The goal of the attacker is to relay the information between Alice and Bob in order to get some illegitimate benefit, even if the communication channel is encrypted.

These attacks can be carried out against most of the existing communication systems, where two parties (Alice and Bob) cannot control the entire communication channel between them. One of the first descriptions of the relay attack was made by Conway [1], referring to a little girl who played chess simultaneously against two Chess Grandmasters. She had to relay the moves from one of the Grandmasters to the other, so that in the end she either won against one of them or drew against both.

As a practical example, think about the situation where Alice is an ATM and Bob is a bank debit card. An attacker can try to mount the following attack to get money from Bob's account. Firstly the attacker creates a fake ATM, which we name MalvAlice (MA). Then he uses a fake card, named MalBob (MB), connected to MalvAlice. Now all the attacker has to do is to convince the possessor of the debit card to insert Bob into MalvAlice. At this point the attacker inserts MalBob into the real ATM, Alice, and relays the communication between Bob and Alice. Both Alice and Bob believe they are communicating directly since they are receiving th correct information; this works even if the data is encrypted, since the only goal of the attacker is to transfer the data between the two parties. At the end the real ATM happily gives the money to MalBob and MalvAlice simply outputs an error message to Bob. A graphical representation is shown in figure 1.

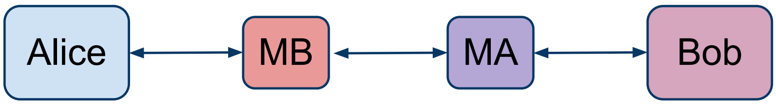


Fig. 1. An example of a relay attack, where a pair of devices forward the communication between Alice and Bob.

Some solutions have been proposed to solve this problem, among which we mention distance bounded protocols [3] and multichannel protocols [6]. The details and differences of these methods compared to our solution are discussed later in the paper, but for now we mention that in this paper we describe a new method to detect relay attacks.

In the following section we describe our idea to detect a relay attack in a more practical way.

3 Our solution: hop-count metric by introducing noise

The core of our idea is to exploit the noise on the channel between Alice and Bob: if the honest participants know how much noise there is on the channel, and if the middlemen cannot avoid introducing extra noise, then the honest participants can detect whether the middlemen are present or not. You can think of our solution as a kind of metric. The more noise received by Bob means the more steps between Alice and Bob.

This is the main intuition, but of course it wouldn't work straight away because the middlemen could use better equipment and better encodings in order to minimize the noise they introduce. So we develop a protocol in which the participants are *required* to introduce some established amount of noise—and it is possible to detect if they don't.

Now of course this wouldn't work either, because the middlemen could reproduce exactly the noise introduced by the honest participants and instead use a perfect noiseless channel between themselves for the relay. So we structure our system so as to make it impossible for the attackers to observe the inputs of the honest participants.

The situation is as follows. Honest participants Alice and Bob already share a secret. Alice has a communication channel with an entity that claims to be Bob. She wants to know whether that entity is really Bob or a middleperson MalBob who pretends to be Bob and relays the answers of the real Bob (in turn tricked by MalBob's accomplice MalvAlice).

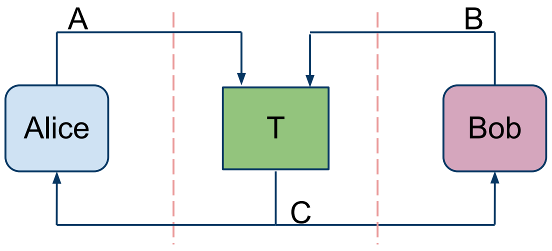


Fig. 2. Diagram of the communication from Alice to Bob when no relay attack is taking place.

The verification works as follows. Alice generates a random bit sequence and transmits it to Bob. Bob is required to add a certain proportion of noise to the channel during the transmission, meaning that he will not receive the exact sequence that Alice transmitted. Alice monitors the channel so she can tell whether Bob is adding the required amount of noise: if he isn't, then he is cheating and she can detect it. Bob is then required to send back to Alice the sequence he received (errors included) and to protect the integrity of the sequence with the secret he shares with Alice. Alice then checks how many errors there are in what Bob said he received. If there are more errors than expected, this indicates the presence of middlemen.

A representation of the communication channel is shown in figure 2. The T box is a trusted entity (perhaps the channel itself) that ensures all participants follow the rules described above. An illustration of a relay attack within this framework is shown in figure 3.

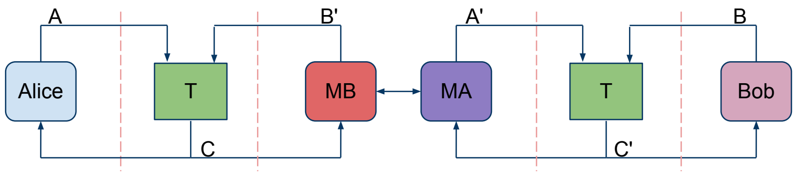


Fig. 3. Diagram of the communication from Alice to Bob when a relay attack is taking place.

One important observation is that if Alice is able to detect all the errors introduced by Bob then MalvAlice can do the same and therefore replicate the errors exactly to Alice. Therefore the protocol must be designed such that some of the errors introduced by Bob are not detected by Alice (e.g. by colliding with her own inputs). This ensures that Alice can detect some of the errors introduced by Bob (where Bob will not be able to find Alice's input), while she will not be able to replicate them all (where Alice cannot find Bob's input).

Another key observation is that Bob must not insert errors on all of the bits from Alice, since this will cause him to receive a sequence independent of Alice's input and therefore would make the detection of a relay attack impossible.

Based on all these requirements we can extract the main features of our desired protocol:

1. Alice and Bob share a common secret which they use for confidentiality and integrity of the data sent between them.
2. Bob must introduce errors; some of them must be detectable by Alice while the others must be indistinguishable to Alice from her own input.
3. For the cases where Bob introduces errors he must be incapable to determine Alice's input.
4. A relay attack should be detectable in all the cases.

With this in mind, we ask ourselves: can such a protocol be implemented in practice? If so, what kind of communication channel can we use?

In the next section we provide two example implementations. These examples have several limitations that make them less suitable for a practical solution but they provide a good example of how to use our methods. Perhaps in the future we can find a more practical realisation.

4 Implementation

In this section we provide two ideal examples of implementation where the T box is the channel itself, with the intention that it therefore can't be bypassed. This channel is a bidirectional bus line that can be driven by both ends as the communication link between Alice and Bob. Such channels are used in many applications, including the banking Chip and PIN system for the communication between a smartcard and a terminal (ISO 7816).

4.1 Method 1

Our first example of implementation uses the following protocol.

Alice sends N bits, one by one by driving the line to 0 or 1 for each bit. The T box connecting Alice and Bob performs a logical AND function. On each bit, Bob must send either 1 (listen to Alice's bit) or 0 (reset the output without reading Alice's bit). Bob is required to send a 0 (reset without reading, i.e. introduce an error) with probability $p = 0.5$ (approximately in half of the slots). If MalBob is in the middle and decides not to reset Alice's input (so that he can forward the exact N bits through MalvAlice to Bob) or to reset every bit (so that Bob will return a correct string of all 0's), then Alice will notice that her correspondent is not following the rules. Therefore MalvAlice cannot replicate Alice's sequence because MalBob cannot observe it (the Alice bits where MalBob drove the line to 0 are hidden).

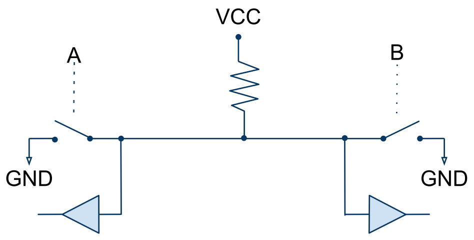


Fig. 4. Diagram of the T box implementation for the first method. A and B can send a 0 by connecting the line to ground or a 1 by leaving it floating. If either participant forces a 0, the result on the line will be 0 regardless of the other's input; otherwise the line will be pulled up to 1 by the resistor.

Remember, from the design requirements, that Alice should detect some of the cases when Bob introduces errors. In this method Alice can detect when Bob

row	**Alice**	**MB**	**C**	**MA**	**Bob**	**C'**	**Outcome**
1	0	0	0	0	0	0	OK
2	0	0	0	0	1	0	OK
3	0	1	0	0	0	0	OK
4	0	1	0	0	1	0	OK
5	1	0	0	0	0	0	Increase error counter
6	1	0	0	0	1	0	Increase error counter
7	1	1	1	1	0	0	attack detected
8	1	1	1	1	1	1	OK

Table 1. Possible outcomes for method 1 from Alice's perspective in the case of a relay attack (see figure 3). When Alice and MalBob send 1 but Bob sends 0 (row 7), Alice detects a relay attack. When Alice observes an error (rows 5 and 6) she increases the observed error counter.

inserts an error (i.e. Bob sends 0) for the cases she sent 1 (since the output is different than her input); however Alice cannot distinguish between Bob sending 0 or 1 for the cases where she sends 0, since her input is masking his.

At the end of the protocol Bob sends to Alice, over the integrity-protected channel, all N bits he received, without Alice interfering. Alice compares this with what she observed on the line. If the pair of attackers MalvAlice and MalBob are involved then for every sequence where Alice and MalBob observe a 1 (they both send a 1) there is a 0.5 probability that Bob will get a different bit on the channel between him and MalvAlice (when he sends a 0 although MalvAlice has forwarded the 1 from Malbob). Alice will notice the difference, detect the men in the middle and refuse to proceed with the transaction. Please refer to table 1 for an illustration of the possible detection scenarios (excluding the absurd cases where the attackers introduce more errors than needed). The attackers cannot avoid the probability of detection. If MalBob inserts too many errors, Alice notices because the counter increased in rows 5 and 6 reaches a higher value than expected; if he inserts too few errors, or even just the prescribed amount of errors, some of the time (one quarter of the cases in which he sends a 1 and thus inserts no errors: rows 3, 4, 7, 8) he'll fall into the situation of row 7 and will be detected immediately. In section 4.3 we describe in detail the probability of detection.

We implement the T box as a bidirectional data line between Alice and Bob, where each participant can either send a 0 by connecting the line to ground, or a 1 by leaving it disconnected and allowing the pull-up resistor to pull it up to VCC (see figure 4). Therefore if any of the two parties forces a 0 the resulting bit on the line (independent of the other) is a 0. This line achieves our goals, assuming "ideal" 0 and 1 logic voltages and under the restriction that both participants can only read and write once per bit time.

4.2 Method 2

In this method we also implement the T box by means of a bidirectional channel (see figure 5), but this time Alice and Bob can select between connecting the line to GND (to send a 0), VCC (to send a 1), or leaving it in a high-impedance mode (state Z, i.e. listen). If either of them decides to listen then the output of the line is equal to the input of the other party. If both of them listen, the line is pulled up to 1 by the resistor. In the case where Alice sends 0 and Bob sends 1 or vice versa we get a short-circuit; since Alice and Bob will never send complementary values if they follow the rules, we take a short circuit as a sure indication of a relay attack.

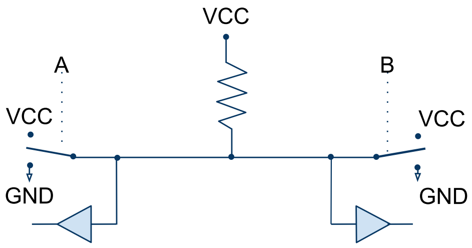


Fig. 5. Diagram of the T box implementation for the second method. A and B can send 0 (connect to ground), 1 (connect to power supply) or leave the line floating (Z) to listen. If both parties select the same non-Z value, the line is driven to that value. If both parties select Z, the line is pulled to 1 by the resistor. If one party selects Z, the result on the line is determined by the input of the other participant. Complementary inputs (0, 1) or (1, 0) create a short-circuit.

The protocol works as follows. First Alice and Bob agree on a sequence of N bits based on their shared secret. Then for N consecutive bit-time slots they choose randomly (i.e. with probability 0.5) either to listen or to send the corresponding bit from the agreed sequence. For the slots where Alice decides to listen (approx. $\frac{N}{2}$), Bob can either listen as well (thus in $\frac{N}{4}$ of the slots) or send the bit from their known sequence ($\frac{N}{4}$ of the times). In the other $\frac{N}{2}$ slots, where Alice sends the corresponding bit from their known sequence, the result on the line is exactly this bit regardless of Bob's input (according to the rules of this protocol, he can only send the same bit or listen—otherwise he will create a short-circuit). As a result, Alice and Bob both leave their output in the high-impedance state (Z) for only $\frac{N}{4}$ of the slots (where they both decided to listen). They will suspect a relay attack if their counterpart appears to be listening more frequently than this (see the next section for more details on how to compute this score).

There is one more problem to solve. In the bidirectional channel we described, a listening Alice cannot really tell whether the other party is listening—he could be sending a 1 and it would look the same. We overcome this problem by considering the case where the bit of the agreed sequence is 0; in this case the correspondent is not supposed to send a 1 and so must be listening, and Alice can count how frequently this happens. And if MalBob won't follow the rules, we detect him even more readily. We show how.

MalBob can either listen or send. If he sends, he must guess the bit of the sequence, because he doesn't know it. With probability 1/2 he will guess it incorrectly. If Alice was also sending in that bit slot, she will observe a short circuit. Honest players never cause a short circuit so Alice will know for sure that it was caused by a relay attack. So, in the case where Alice and MalBob both send, and MalBob guesses incorrectly, the relay attack is detected immediately during that same bit slot, without even having to run the protocol until the end.

On the other hand, if MalBob always listens (Z) in order to avoid short circuits, he will be detected probabilistically by the fact that Alice sees her correspondent listen more frequently than he ought to. But how can Alice reliably detect whether her correspondent is listening?

Alice considers the case in which the bit of the secret common sequence is 0 and she listens. If the correspondent is Bob, half the time he should be listening too (observed value: 1) and half the time he should be sending the correct bit (observed value: 0). If the correspondent is a MalBob who always listens, the observed value will be always 1, and Alice will know that something is wrong because she never sees any of the expected 0s. Interestingly, if MalBob attempts to guess half the time, he'll be perfectly OK *in this case*: if he guesses correctly and sends a 0, it will look to Alice as if Bob sent the right bit; and, if he guesses incorrectly and sends a 1, it will look to Alice as if Bob listened. However, as we said above, on the whole the strategy of guessing is not sustainable for MalBob because he will be detected immediately when he guesses incorrectly in the *other* case, when the bit of the sequence is 1 and Alice is listening—there, the observed outcome with MalBob will be 0, which the genuine Bob would never generate. Since MalBob doesn't know the secret bit, he can't tell the two cases apart and thus can't tell whether he can safely get away with guessing or not.

If MalvAlice and MalBob want to relay the information between Alice and Bob they must trade-off between *guessing* the secret sequence agreed by Alice and Bob or *listening*. In the first case the attackers can be detected immediately as soon as they create a short-circuit or introduce a wrong bit. In the second case, in the long run Alice and Bob can notice that their counterpart is listening more than expected.

With reference to all the possible cases as listed in table 2, to detect whether her correspondent is listening too frequently, Alice assumes he knows the secret sequence (safe in the knowledge that, if he doesn't, he will be discovered in rows 2, 4 or 10 at one of his incorrect guesses) and checks the case in which the secret is 0 and she is listening (third group, rows 7, 8 and 9). She expect to hear a 1 (indicating that Bob is listening, row 9) approximately half the time, and a 0 (indicating that Bob is sending, row 7) the other half. The more the frequency deviates from this, the more she should be suspicious.

Note that row 8 can only be generated by MalBob (by guessing the secret bit incorrectly), never by Bob. Alice can't tell the difference between rows 8 and

row	**Secret**	**Alice**	**Bob or MalBob**	**Line**	**A's conclusion**
1	0	0	0	0	OK (B sent 0 or listened)
2	0	0	1	S	ATTACK DETECTED (short-circuit)
3	0	0	Z	0	OK (B sent 0 or listened)
4	1	1	0	S	ATTACK DETECTED (short circuit)
5	1	1	1	1	OK (B sent 1 or listened)
6	1	1	Z	1	OK (B sent 1 or listened)
7	0	Z	0	0	OK (B sent 0)
8	0	Z	1	1	Increment counter (B listened or cheated)
9	0	Z	Z	1	Increment counter (B listened or cheated)
10	1	Z	0	0	ATTACK DETECTED (wrong secret bit)
11	1	Z	1	1	OK (B sent 1 or listened)
12	1	Z	Z	1	OK (B sent 1 or listened)

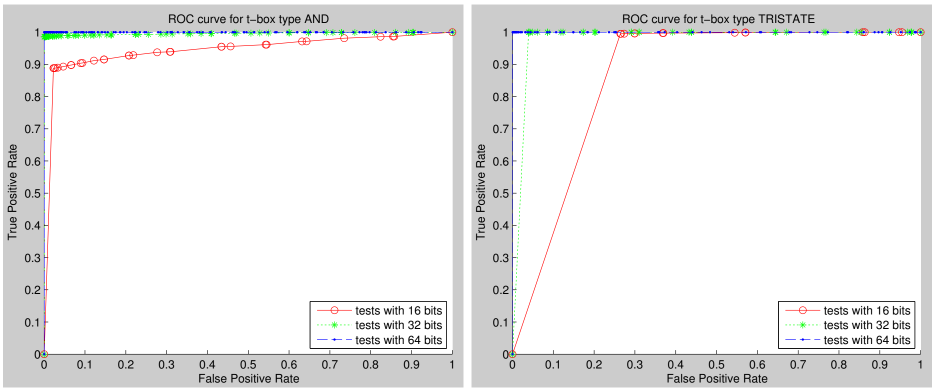
Table 2. All possible outcomes from Alice's perspective in method 2. There are a total of 12 possible combinations, out of which MalBob is detected immediately in 3 (rows 2, 4, 10). There is no case in which Alice can tell for sure that the other party is listening, but in rows 8 and 9 she knows that, if he didn't listen, then he cheated.

9, so this is the only case in which MalBob can guess incorrectly and not be discovered. But Alice doesn't care: if MalBob guesses rather than listening, in 3/4 of the cases in which he guesses incorrectly (thus in 3/8 of the cases in which he guesses) he will be immediately discovered.

Bob, meanwhile, can perform the dual checks from his side to detect the presence of MalvAlice. For lower communication overhead compared to the first method, each of them might just compute a detection score (see next section) locally. But, for higher detection accuracy and fewer false negatives, at the end of the protocol we instead require Alice and Bob to send their local scores to each other over their cryptographically protected channel and then compute a common global score as the maximum of the two local scores.

4.3 Evaluation

There are several differences between the first and second methods. In the first method we require Bob to send the entire sequence of N bits back to Alice in order to detect a relay attack, while on the second method at least one of the participants will spot the attack independently and could immediately end the transaction. A slight difference on the communications side is that, in the second method, what they send back at the end is not the bit sequence, which is pre-agreed, but just the numerical scores; but *both* have to send their respective scores. (In the first method, since it's only Alice who computes the score, after doing so she must send Bob the boolean verdict; so, in the end, the overall communication costs are very similar between the two methods.) A physical difference is that in the second method the participants are required to detect and handle short-circuits without electrical damage, which might require extra hardware.



(a) (b)

Fig. 6. The ROC curves for method 1 (plot (a)) and method 2 (plot (b)). These plots are based on tests with $n = 10000$ protocol runs and different message lengths ($l \in \{16, 32, 64\}$).

Both methods proposed use a probabilistic approach to detect a relay attack. Therefore we have created a Python simulation[1] to test and compare the effectiveness of the two methods. Each method computes a detection probability score ($f \in [0, 1]$) and based on a set threshold ($r \in [0, 1]$) decides whether to signal or not a relay attack detection. The detection probability scores are computed as follows. In the first method, Alice checks if the sequence she observed on the line is the same as the one received from Bob. If they differ she signals a relay attack (with certainty) by setting the detection score to 1. Else, she computes the score based on how many times Bob inserts errors:

$$score_1 = \left|1 - 2 \cdot \frac{seen\_errors}{sent\_ones}\right| \tag{1}$$

where $seen\_errors$ represents the number of times Alice detects an error (she sends 1 but observes 0) and $sent\_ones$ represents the number of 1's she sent. This score, between 0 and 1, is lowest when $seen\_errors = \frac{sent\_ones}{2}$, which is the expected number of errors that Bob should insert, and highest when Alice never sees any errors ($seen\_errors = 0$) or sees errors at each possible occasion ($seen\_errors = sent\_ones$).

In the second method, Alice checks for a short circuit (rows 2 and 4) or a wrong secret bit (row 10) and if one is found she sets the detection score to 1. Else the score is obtained based on how often her correspondent listens:

[1] `http://www.cl.cam.ac.uk/~osc22/projects/tbox_norelay/pyscripts/tchannel.py`

$$score_{2A} = \left| 1 - 2 \cdot \frac{b\_listened\_or\_cheated}{a\_listened\_while\_0} \right| \tag{2}$$

where *b_listened_or_cheated* is the number of times that Alice saw a 1 on the line while the secret bit was 0 and she was listening (rows 8 and 9) and *a_listened_while_0* is the number of times that Alice listened while the secret bit was 0 (rows 7, 8 and 9).

This score, between 0 and 1, is lowest when $b\_listened\_or\_cheated = \frac{a\_listened\_while\_0}{2}$, i.e. when Bob appears to listen in exactly half the cases examined, and highest when he seems never to listen ($b\_listened\_or\_cheated = 0$) or always to listen ($b\_listened\_or\_cheated = a\_listened\_while\_0$).

Bob performs the symmetrical calculation and obtains $score_{2B}$. They then exchange these scores and compute the overall

$$score_2 = \max(score_{2A}, score_{2B}). \tag{3}$$

Using this simulation framework we tested both methods for $n = 10000$ runs of the protocol, on both relay and legitimate scenarios and for different message lengths ($l \in \{16, 32, 64\}$). Based on this data and the algorithm described by Fawcett [9] we have plotted the Receiver Operating Characteristic curves shown in figure 6. From the two plots we can observe that method 2 is slightly better than method 1. With a sufficiently long message, relay attacks are very unlikely to remain undetected. In the $n = 10000$ relay scenario tests of our simulation, all attacks were detected when we used messages of length 64 bits (method 1) and 32 bits (method 2).

4.4 Limitations

The two examples above are meant to convey the flavour of our idea but, as presented, they are only ideal descriptions. Further developments are needed before they can be considered robust against real-world attackers.

First of all, we assumed time to be ideally quantized: we required that each of the participants connected to the bidirectional line (i.e. the T box) can only read and write once per bit slot. In the first method, once MalBob has decided to select a given input, we assumed he cannot change this input. In the second method, once MalBob has decided to listen, we assumed he cannot change his mind and send a value in that bit slot after hearing what Alice's value was. In practice, nothing stops MalBob from doing just that. Perhaps our examples could work if the two honest participants (Alice and Bob) had faster equipment than the attackers. But this creates a linear race between the hardware capabilities of the attackers and the honest participants. A great improvement would be to find an implementation that requires MalBob and MalvAlice to have communication equipment exponentially more powerful than Alice and Bob's.

We also assumed voltage to be ideally quantized, with values being either 0 or 1: but in practice we expect that enterprising attackers will be able to spot a difference in the analog voltage of the line between different combinations of the input. In method 1, for example, if MalBob is sending a 0 and thus the outcome is necessarily 0, he might still be capable of detecting a slight difference between Alice driving her end of the line with a 0 or a 1. There is also the question of who supplies VCC—the attacker might exploit that as a side channel.

Another limitation of our solution is that is vulnerable to the *long cable* problem. This problem refers to a particular case of the relay attack, where the attacker connects Alice and Bob via a longer cable than they would expect, without creating two separate communication channels (see the paper by Clulow et al. [8] for more details). In this scenario there is no extra noise created by MalBob so our method could not detect the attack.

5 Related work

Relay attacks were first discussed by Conway [1], exemplifying a little girl that plays chess simultaneously against two Chess Grandmasters. By forwarding the moves from one of the Grandmasters to the other she is able to win against on of them or draw against both. A similar scenario, oriented towards payment protocols, was described as the *mafia fraud* by Desmedt et al. [2], who first proposed a countermeasure based on measuring the round trip time.

One of the first concrete solutions was proposed by Brands and Chaum, as *Distance-Bounding Protocols* [3]. Hancke and Kuhn proposed a distance-bounded protocol for RFID communications [4], which was later adapted by Drimer and Murdoch for the ISO 7816 (smartcards) scenario [7]. These solutions can provide an estimate of the distance between two communicating parties, based on the fact that a signal cannot travel faster than light. These protocols can provide a resolution of approximately 1.5 m using a dedicated hardware running at 200 MHz, although they only require powerful hardware in one side (e.g. the smartcard reader) while posing little computational constraints on the other participant (e.g. the smartcard).

Our solution, even though it does not yet have a secure implementation and so a direct comparison is unfair, doesn't have any distance limitations: a relay attack can be detected even if the attackers are arbitrarily close to the honest players.

Another type of defense against relay attacks was proposed by Stajano et al. [6] using multichannel protocols. The proposed protocols require that the two communicating parties use a digital channel for cryptographically secure data communication and a physical channel (e.g. a banknote that can be destroyed) that cannot be relayed. But the scheme does not yet have a practically usable implementation.

The *mafia fraud* kind of relay attacks, such as the demonstration on the Chip and PIN system by Drimer and Murdoch [7], could be blocked using an electronic attorney device [10]. The first author has created such device [5], which contains a trusted user interface and can detect if a fake reader is showing a false amount. However this solution cannot protect against similar attacks where the bandits are providing the expected request (e.g. withdraw money from ATM), which the more general solution proposed in this paper would instead detect.

6 Conclusions and further work

In this paper we have described a new idea to detect relay attacks. Such attacks are possible against most systems, since the attackers do not require any knowledge about the underlying protocol or cryptographic keys.

Our approach is based on introducing intentional errors (noise) between two communicating parties that share a common secret for confidentiality and integrity of the channel (whisper). We have designed our solution such that any pair of attackers trying to execute a relay attack must introduce extra errors in the communication and this should be detected by the honest participants.

In order to illustrate our idea we have described two examples of implementation and we have evaluated their detection performance using a simulation framework. However these implementations are idealized under several respects and, as presented, would not withstand a real-world analog attacker. We believe however that our idea offers a new strategy for detecting relay attacks and we hope that a more practical implementation may be found in the future.

7 Acknowledgements

We thank Markus Kuhn for helpful discussions and for pointing out the vulnerability against the *long cable* problem. Also we thank Mike Bond for many comments about the earlier drafts of the paper as well as for the initiative with the simulation framework.

Omar Choudary is a recipient of the Google Europe Fellowship in Mobile Security, and this research is supported in part by this Google Fellowship. The opinions expressed in this paper do not necessarily represent the views of Google.

References

1. J. Conway, "On numbers and games, pp. 75". *Academic Press*, 1976.
2. Y. Desmedt and C. Goutier and S. Bengio, "Special Uses and Abuses of the Fiat-Shamir Passport Protocol". *Advances in Cryptology, CRYPTO '87.*
3. S. Brands and D. Chaum, "Distance-Bounding Protocols". *Proc EUROCRYPT 93*, LNCS 765.
4. G. Hancke and M. Kuhn, "An RFID Distance Bounding Protocol". *Proc IEEE Securecomm '05.*.
5. "The Smart Card Detective: a hand-held EMV interceptor", Omar Choudary, MPhil thesis at University of Cambridge, Computer Lab, `http://www.cl.cam.ac.uk/~osc22/scd/`.
6. Frank Stajano, Ford-Long Wong and Bruce Christianson, "Multichannel protocols to prevent relay attacks". *In proceedings of Financial Cryptography 2010*, Springer LNCS.

7. Saar Drimer and Steven Murdoch, “Keep your enemies close: distance bounding against smartcard relay attacks”. *16th USENIX Security Symposium*, 8/2007.
8. Jolyon Clulow, Gerhard P. Hancke, Markus G. Kuhn and Tyler Moore, “So Near and Yet So Far: Distance-Bounding Attacks in Wireless Networks”. *European Workshop on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS)*, Hamburg, Germany, September 2006, LNCS 4357.
9. Tom Fawcett, “ROC Graphs: Notes and Practical Considerations for Researchers”. *Kluwer Academic Publishers*, Netherlands, 2004.
10. Ross Anderson and Mike Bond, “The Man-in-the-Middle Defence”. *Cambridge Security Protocols Workshop*, 2006.