The Not-so-Distant Future: Distance-bounding Protocols on Smartphones

Sébastien Gambs[1], Carlos Eduardo Rosar Kós Lassance[2], and Cristina Onete[3]

[1] Université de Rennes 1 - Inria / IRISA
sgambs@irisa.fr
[2] Université de Rennes 1 / Télécom Bretagne
cadurosar@gmail.com
[3] Inria / INSA Rennes
cristina.onete@gmail.com

Abstract. In authentication protocols, a relay attack allows an adversary to impersonate a legitimate prover, possibly located far away from a verifier, by simply forwarding messages between these two entities. The effectiveness of such attacks has been demonstrated in practice in many environments, such as ISO 14443-compliant smartcards and car-locking mechanisms. Distance-bounding (DB) protocols, which enable the verifier to check his proximity to the prover, are a promising countermeasure against relay attacks. In such protocols, the verifier measures the time elapsed between sending a challenge and receiving the associated response of the prover to estimate their proximity. So far, distance bounding has remained mainly a theoretical concept. Indeed in practice, only three ISO 14443-compliant implementations exist: two proprietary smartcard ones and one on highly-customized hardware. In this paper, we demonstrate a proof-of-concept implementation of the Swiss-Knife DB protocol on smartphones running in RFID-emulation mode. To our best knowledge, this is the first time that such an implementation has been performed. Our experimental results are encouraging as they show that relay attacks introducing more than 1.5 ms are directly detectable (in general off-the-shelf relay attacks introduce at least 10 ms of delay). We also leverage on the full power of the ISO-DEP specification to implement the same protocol with 8-bit challenges and responses, thus reaching a better security level per execution without increasing the possibility of relay attacks. The analysis of our results leads to new promising research directions in the area of distance bounding.

Keywords: Distance-bounding protocol, relay attack, NFC, smartphone.

1 Introduction

Radio Frequency IDentification (RFID) is a widespread and cost-efficient technology, currently used for applications ranging from contactless payment to public transport and machine-readable identification. A fundamental cryptographic

The authors are listed by alphabetical order.

primitive that must be supported and implemented on RFID technology is *authentication*. During an authentication protocol, the device equipped with an RFID chip (also called the *prover*) interacts with a reader (called the *verifier*) to prove its legitimacy. To protect the privacy of the data stored and manipulated by RFID tags for sensitive applications, these devices must be equipped with a processor capable of performing cryptographic operations.

RFID chips may be embedded in devices such as passive tags securing items in supermarket, public transport cards as well as contactless payment smartcards. Thus, the term "RFID prover" covers a wide range of hardware, which differs in terms of characteristics such as memory, surface area, independence with respect to the verifier for energy (*i.e.*, passive versus active) and cost. However, traditional RFID provers always answer requests from the reader without asking for the user's consent. This property makes RFID technology prone to *relay attacks*. These attacks are known as mafia frauds [5] in the context of authentication, or wormhole attacks in the field of neighbourhood discovery. In a relay attack an adversary impersonates a legitimate prover by forwarding messages between him and the legitimate verifier. Relay attacks have been successfully implemented against Bluetooth [9], ISO 14443 smartcards [3,11], electronic passports [13], electronic voting schemes [19], and even access mechanisms for cars, such as Passive Keyless Entry and Start (PKES) [7].

To counter relay attacks, Brands and Chaum introduced *distance-bounding* (DB) protocols [2]. A DB protocol extends classical authentication schemes by additionally enabling the verifier to check his proximity to the prover. Most RFID tags operate according to the ISO 14443 standard, for which "proximity" is defined in the range of 10 cm. In a DB protocol, the verifier is equipped with a clock, which measures the roundtrip time of fast challenge/response rounds. As RFID communication takes place at the speed of light, such time measurements accurately reflect the communication distance, and thus can be used for proximity checking.

While the literature of DB protocols is abundant (see for instance [1,2,10,16,6]), to the best of our knowledge only three practical implementations of RFID distance bounding currently exist. The first one is a highly-customized (and expensive) proof-of-concept implementation by Ranganathan, Tippenhauer, Singelée, Škorić and Capkun [20]. The two other solutions, namely the Proximity Check option for NXP's Mifare Plus cards and the solution of 3DB Technologies, come from industry and run with proprietary specifications and hardware.

Modern smartphones can use Near Field Communication (NFC) technology to emulate RFID tags. In this mode, the phone can behave either as an RFID prover or as a verifier, and its communication is subject to the ISO-DEP protocol specified in the ISO14443-4 standard [14,18]. Smartphones have already been used by Francis, Hancke, Mayes and Markantonakis to conduct relay attacks [8]. Their work demonstrates that any application, ranging from credit cards to electronic passports, can be attacked by using NFC phones, even if the relay attack introduced a delay of several seconds. In terms of distance, this allows the relay attacker to impersonate a prover located several hundreds of thousands

of kilometers away from the verifier. However, to our best knowledge no DB protocols have so far been implemented on smartphones.

Main contributions. In this work, we investigate the implementability of DB protocols on smartphones running in a tag-emulation mode, focusing in particular on relay attacks and mafia frauds using off-the-shelf hardware. More specifically, our countermeasures cannot compete with fast tailored hardware that can perform relays in some microseconds. Moreover, we do not consider attacks such as distance and terrorist frauds.

More precisely, we explore how DB can be implemented on Android *without* changing the design of the hardware (such as the SIM card, the phone's processor or the phone itself). In contrast to the proximity check solutions of Mifare Plus and 3DB Technologies, our implementations are public and do not require proprietary hardware. Our main objective is to evaluate how far one can implement DB protocols as a standard Android application on an existing smartphone. We do not rely on the SIM card as a cryptographic processor, nor customize its properties. Rather, we directly use the processor of the smartphone. As a consequence, our implementation is easier to adopt, but prone to side-channel and malware attacks.

We propose three proof-of-concept implementations of increasing complexity of the Swiss-Knife DB protocol [16] on Android. All these three implementations use 32 challenge-response rounds.

1. **The basic implementation.** This first implementation works at the application layer and does not require root access. While it can be used straightforwardly, our results show that the roundtrip time measurement presents a lot of variation, with a standard deviation of 4.4 ms for a distribution containing multiple clusters of values.
2. **The customized implementation.** In this second implementation, we modify the Android operating system to make the protocol run at the Hardware Abstraction Layer (HAL). We customize both the prover and the verifier to decrease the processing delays introduced by them, and reduce variations caused by propagating messages across all the layers. Our results show a normal distribution of measured roundtrip values of this implementation, with no false negatives when the protocol is implemented with an error threshold of 16 rounds and an allowed variation of 1.5 ms from the minimal observed measurement.
3. **The 8-bit implementation.** The phone follows the ISO 14443-4 standard and the ISO-DEP protocol for RFID emulation, which encodes the 1-bit challenges of the Swiss-Knife protocol as an entire byte. For our third implementation, we extended our second implementation by using the transmitted byte to exchange 8-bit challenges and responses. Our results are encouraging as the measured times present little variation while the gain in security level per execution is visible even for a proximity threshold within 1 ms of the minimum time. This last result indicates that designing DB protocols using larger challenges is a promising direction, in particular since the encoding of the message according to the ISO-DEP protocol *must* be at least the size

of a byte. This contradicts the theoretical design recommendations for DB protocols outlined in a previous work by Clulow, Hancke, Kuhn and Moore [4].

Our analysis shows how to choose the parameters for the proximity bound and for the threshold value of erroneous transmissions. In particular, choosing a proximity bound within 1.5 ms of the observed minimum time is optimal for our second implementation and reasonably good for our third implementation. By contrast, the first implementation requires a much larger variation. In terms of fault tolerance and the resulting security level, the third implementation largely outperforms the two other ones. Indeed, even when 20 out of 32 rounds are faulty, this implementation still ensures a better security level than the second implementation with 32 out of 32 rounds being correct.

We also discuss the somewhat negative result that our experimental implementations cannot detect relays at less than 1 ms if the threshold t_{max} is chosen as to provide a good tradeoff between correctness and relay-attack detection. However, all practical relay attacks necessarily introduce delays. For fast dedicated relay hardware, such as that of Hancke [11], the relay is in the order of micro-seconds and thus undetectable. However, other previous attacks such as [8] and [12] introduced delays of respectively 50 and 27 milliseconds, which can be easily detected by our implementation. Thus, our results are encouraging in the sense that the proximity check can only be bypassed by a Man-In-the-Middle (MIM) adversary using more sophisticated technology than standard off-the-shelf mechanisms.

Outline. The paper is organized as follows. In Section 2, we review some preliminaries on relay attacks, DB protocols and relevant Android/RFID-emulation details. Afterwards in Section 3, we describe our methodology, before discussing our results in Section 4. Finally, in Section 5 we conclude by putting our work in perspective and outline future research directions.

2 Preliminaries

Relay attack. An authentication protocol conducted between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ aims to allow the latter party to check the *legitimacy* of the former, as a prerequisite of allowing $\mathcal{P}$ access to specific privileges (*e.g.*, entering a building, using a public transportation system or unlocking a car).

In a *relay attack*, an adversary $\mathcal{A}$ authenticates as legitimate by relaying messages between an honest verifier and an honest prover located far from the verifier. The adversary "extracts" honest authentication responses from the prover by means of a *Leech* – a device acting as a verifier – and authenticates by using a *Ghost* – a second device, acting as a prover and extracting information from the verifier. If the attack is successful, then the Ghost authenticates. This attack is illustrated in Figure 1.

Relay attacks are possible in practice and bypass usual cryptographic countermeasures like encryption and digital signatures. They are even more effective

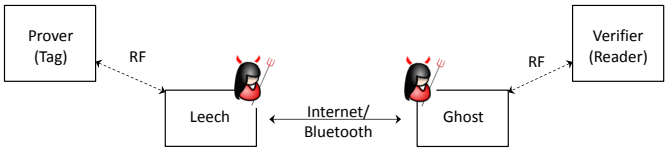

Fig. 1. A relay attack: the Ghost tries to authenticate to an honest verifier by forwarding the information received by the Leech from the honest prover.

in the context of RFID authentication, since RFID tags do not require the consent of the user to transmit messages. Such tags usually require the proximity of the verifier to transmit (being otherwise passive). However, relay attacks can contradict this assumption by "amplifying" signals (*e.g.*, using a stronger magnetic field) and by forwarding messages even from very distant provers. As a consequence, an adversary can use a far-away user's card to enter a sensitive area, or make a contactless-card payment by using a card located hundred of kilometers away from the terminal [15].

Distance-bounding (DB) protocols. In this work, our main focus is on implementing DB protocols on smartphones to be able to detect relay attacks (*i.e*, mafia frauds). We do not address the two other attacks usually considered in the DB literature, namely (1) the *terrorist fraud*, in which a Man-In-the-Middle adversary receives limited one-time help from a malicious prover to authenticate and (2) the *distance fraud*, in which a malicious far-away prover makes the verifier believe he is actually in his proximity.

In general, DB protocols start with an *initialization* phase, during which several protocol parameters and nonces are exchanged. Then, during the *interactive* phase, the prover and verifier run several fast rounds in which $\mathcal{V}$ first sends a one-bit challenge to which $\mathcal{P}$ responds with a single bit answer. The verifier stores the response and the measured roundtrip time. Finally, during the *verification* phase, $\mathcal{V}$ checks the responses sent during the fast exchanges and compares the roundtrip times to a threshold t_{max}. Based on the results obtained, the verifier authenticates (or not) the prover.

We implemented the well-known Swiss-Knife DB protocol [16], which we briefly present hereafter. In this protocol, each prover $\mathcal{P}$ is associated with an identity ID, stored by the verifier $\mathcal{V}$ together with a shared secret key sk per prover. As depicted in Figure 2, in the initialization phase of this protocol, the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ exchange two session-specific nonces $N_{\mathcal{P}}$ and $N_{\mathcal{V}}$. Then, $N_{\mathcal{P}}$ and the key sk are used to derive two bitstrings T^0 and T^1 later used as responses in the N_c interactive challenge-response rounds. The former value corresponds to the output of a pseudo-random function (PRF) keyed with the shared key sk and given as input $N_{\mathcal{P}}$ (*e.g.*, usually this function is simply a HMAC). The second bitstring T^1 is computed as the XOR of T^0 and sk.

In each challenge-response round, $\mathcal{V}$ generates a random 1-bit challenge, and $\mathcal{P}$ responds with a bit from either T^0 or T^1 depending on the challenge. In the verification phase, $\mathcal{P}$ authenticates the transcript by again using a PRF. Finally, $\mathcal{V}$ authenticates $\mathcal{P}$ if and only if: (1) the responses are correct, (2) the validation of the transcript by the prover succeeds and (3) a certain number of rounds have arrived on time. Indeed, the Swiss-Knife protocol tolerates some faulty/untimely challenge-response rounds up to a predefined threshold E_{MAX}. Previous provable-security analyses [16,6] show that the Swiss-Knife protocol run with a total of N_c fast challenge-response rounds has an (approximate) mafia-fraud resistance of $2^{-(N_c - E_{\mathsf{MAX}})} + \epsilon_{\mathsf{PRF}}$, in which ϵ_{PRF} is the advantage an optimal distinguisher has in distinguishing the output of PRF from a truly random string.

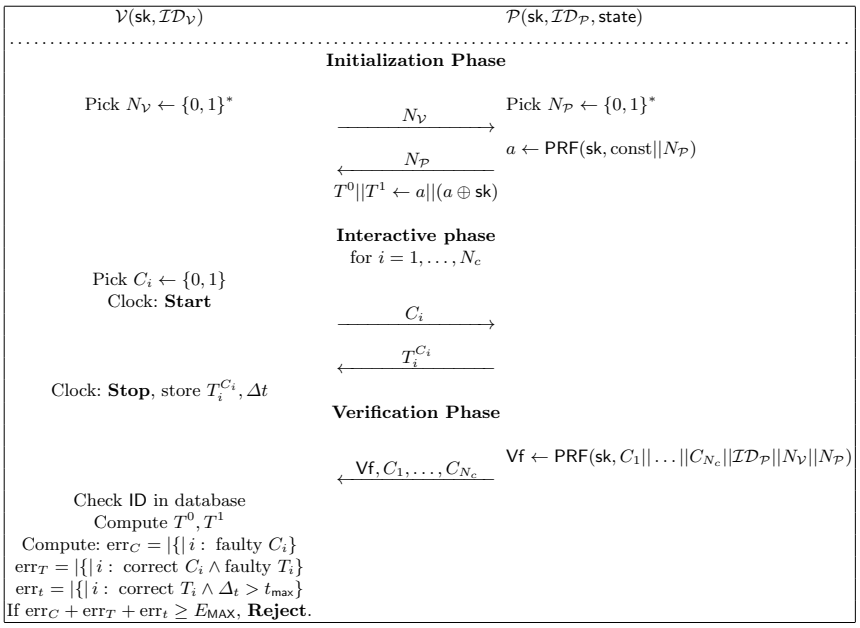

Fig. 2. The Swiss-Knife protocol with unilateral authentication.

NFC communications on Android. NFC communications on Android smartphones are characterized by two elements: the host (the Android operating system) and the controller (usually the NFC chip communicating internally with the host and externally with other chips). The core of the NFC communications depends on the type of controller existing in the phone. For older smartphones, the controller was designed by NXP and its implementation uses the libnfc-nxp library, relying on two protocols: LLC (Logic Link Control) and HCI (Host Controller Interface). More recent controllers follow the NFC Controller Interface

implementation (NCI) and use the libnfc-nci library [17]. The specific protocols used by each controller are highly relevant as they usually introduce significant delays and encode the transmissions in a way that decreases the useful information rate. Thus, in the context of distance-bounding such techniques render proximity checking more difficult.

RFID communication on a smartphone occurs via a serial port with a baud rate of 115200 using an 8-N-1 configuration. This yields $11520 * 8$ information symbols per second, which is equivalent to a constant rate of 11520 bytes per second. Thus, transferring a byte of information will take approximately 0.0868 ms. Two controllers can communicate via a peer-to-peer protocol or they can behave as reader and tag communicating via the ISO-DEP protocol [14,18] summarized in Figure 3. We chose to implement this last option in which the controller of the prover emulates an ISO/IEC 14443-4-compliant passive NFC-A tag. For the predefined carrier frequency of 13.56 MHz, the bit duration according to the ISO/IEC 14443 protocol is 128/13.56MHz = 9.439 microseconds, or 75.512 microseconds per byte.

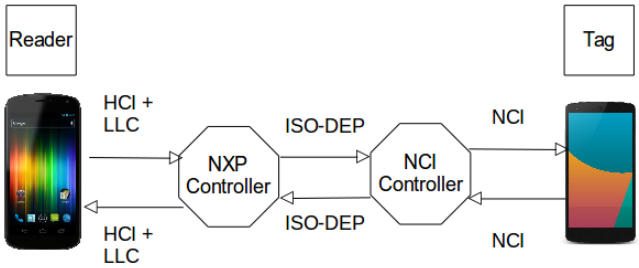

Fig. 3. Phone-to-phone communication between an NFC prover and an NCF verifier.

Older phones generally use the LLC protocol on the outer interface and the HCI protocol for the inner communication. In more recent phones, the NCI protocol functions as a token-based mechanism, which can be invoked by the controller to eliminate buffer overflows [17]. In this setting, the controller gives so-called *credits* to the host that can be used to communicate. Finally, the ISO-DEP protocol defines the format of frames exchanged between the two controllers. This format divides a frame in three blocks, each consisting of 2 bytes: the Start of Data (SoD), the Payload and the End of Data (EoD). At a per-byte duration of 75.512 μs, it takes 0.151 ms to transfer each byte from the host to the controller and back. In addition to each payload byte, the SoD and EoD blocks amount to 4 overhead bytes (*i.e.*, a 0.604 ms delay), resulting in a minimum communication time of 0.755 ms.

On Android, the data transmitted by an application to the chip must pass through all the abstraction layers provided by the operating system, from the

more abstract to the more concrete: (1) the application layer, (2) the Android NFC-Service layer, (3) the Android NFC Application layer, (4) the Libnfc NXP/NCI layer, (5) the Hardware Abstraction Layer (HAL), (6) the driver/serial port, and finally (7) the NFC card layer. These abstraction layers exist to facilitate the work of the developer and to enable an easy integration of various types of hardware within Android. However, each layer also adds delays to the transmission time for both sending and receiving data. In our experiments, this delay was at least 1.5 ms for each smartphone. Circumventing these layers require root access in order to change the Android operating system or at least to install and use modified Android libraries.

3 Implementing Distance Bounding on Android

In our implementations, the verifier is a Samsung Galaxy Nexus smartphone following NXP specifications for the communication with its NFC chip. The prover (*i.e.*, tag) is implemented on an LG Google Nexus 5 running Android Lollipop (5.1) and using NCI specifications for phone-to-chip communication. The Swiss-Knife protocol was run with $N_c = 32$ challenge-response rounds and a variable tolerance threshold E_{MAX} for the maximum number of errors it can accept. An error can be a faulty challenge/response or a transmission time surpassing the threshold t_{max}. For our implementations, the threshold E_{MAX} is essential and a trade-off has to be set between reducing it and increasing t_{max} (thus permitting some relay attacks due to the spread of the response times).

Our experiments were run in 10 batches of 20 executions each, the batches themselves being separated by a random interval of time. For the computation of the pseudorandom function PRF, we relied on the implementations of HMAC_SHA1 or HMAC_SHA256 as explained below. We implemented three versions of the reader and tag. The first runs the protocol with no modification to the Android operation system (*i.e.*, the phone is used in a "normal" mode). The second one executes the protocol by changing the Android operation system so as to bypass the stack layers in the roundtrip-time estimation. Thus, we call it the "customized" implementation. Finally, the third version modifies the structure of the Swiss-Knife protocol by using 8-bit challenges and responses. Hereafter, we use the following notation for the different implementations: challengesize : verifiermode $\leftrightarrow$ provermode, in which the prover and verifier modes are always either normal or custom, and the challenge size is either 1 or 8 bits. Thus, 1 : normal $\leftrightarrow$ custom denotes the implementation with 1-bit challenges (as specified by the Swiss-Knife protocol), with the prover in normal mode and the verifier in custom mode.

For the custom mode, we modify the Android operating system by using the sources available on the AOSP (Android Open Source Project)[1]. For the reader protocol, we used the branch called android-4.2.2_r1, while the tag protocol uses

[1] These sources can be downloaded at `https://source.android.com/source/downloading.html`.

the android-5.1.0_r3 branch. For our normal mode implementations, the reader uses the *"LMY47I"* stock version and the tag uses the *JDQ39* version[2].

When the reader runs in custom mode, it uses a fully customized Android image. In contrast, since the prover protocol is meant to be used by everyday users, we limited the tag customization to substituting the default library nfc_nci.bcm2079x.default.so with our own customized version of it. This library adds the Swiss-Knife implementation to the logic of the HAL library. To simulate a regular user environment for the prover, the smartphone implementing the tag has a number of applications installed on it and is always connected to a 3G/4G network.

The fast challenge/response roundtrip time is measured starting from the moment the reader activates its sending function to the time it receives the size of the response packet. If this information was sent as soon as it was computed, then measuring the time of arrival of this first part can provide an accurate proximity estimation. However, the driver of the phone acting as a tag actually buffers all the response data before sending it to the HAL, which leads to an overhead corresponding to the full processing and computation time for the response. An additional limit is due to the granularity of the reader's clock, which can only measure with a precision of up to 30.518 μs or 5 km. While this granularity is incompatible with the precision generally required by a DB protocol, it is sufficient for our scenario, since most challenge-response time measurements are in the order of several ms.

Note that changing the models of the smartphones used could lead in a slight variation in measured times. In particular, changing the reader phone to a newer model could produce more accurate results, while changing the tag may lead to a choice of the t_{max} bound allowing a MIM adversary less time to perform a relay. Our decision of implementing the DB protocol of the phone itself, rather than on the SIM card, makes it easier to install and deploy at large scale. However, consequently our implementation is also less efficient and secure.

3.1 1 : normal ↔ normal

For the first implementation, we created a pair of Android applications (one for the prover and the other for the verifier), which execute the Swiss-Knife protocol using the development tools found in `https://developer.android.com/sdk/index.html`. Here, our primary objective was to implement the protocol in the simplest way possible. This implementation exchanges the necessary bytes via NFC using the standard Android API for NFC communications. This application stores the measurement obtained for each of the $N_c = 32$ challenge-response rounds. The length of the variables $N_{\mathcal{V}}$, $N_{\mathcal{P}}$, sk, the constant const and of the verification string Vf is 32 bits. The (full) challenge string and each of the two response ones are also of the same size. We rely on the HMAC_SHA1 implementation from the package javax.crypto for our PRF. This function outputs a

[2] These versions are available at `https://developers.google.com/android/nexus/images`.

160-bit string, which we truncate to 32 bits. The values of sk and the constant const are generated at random and saved on the phone.

The results of the 200 measurements (in 10 batches of 20 executions) are summarized in Table 1. We recall that our main objectives were: (a) to prevent relay attacks by choosing a tight value of t_{max}, (b) to ensure correctness, thus setting a trade-off between the tightness of t_{max} and the value of E_{MAX} and (c) preventing mafia fraud by optimizing the choice of E_{MAX}. In particular, we did not consider attacks such as terrorist and distance frauds. We also did not investigate the delay induced by each abstraction layer as it would require changing the operational system in the so-called "normal" implementations, possibly altering the results due to the measurement. For the first implementation, we observed an important variation in the measured times, which range from 9.37 to 82.0 ms, with a standard deviation of 4.4 ms (which is almost half our minimum time). In practice, this means that in order to optimize correctness (objective (b)), we need either a very high value of t_{max}, which allows practical relays to be mounted, or a very high value of E_{MAX}, which would lower the mafia fraud resistance. The median time of 18.95 ms indicates that at least half the measurements are more than twice this minimal value. Thus, even having an error margin E_{MAX} as high as $\frac{1}{2}N_c = 16$ rounds still results in the adversary having a 9 ms window for mounting an attack.

3.2 1 : custom ↔ custom

For our second implementation, we customized the implementation by short-circuiting some abstraction layers in the phone, which creates a trust issue. In particular, one must ensure that only NFC transmissions are spoofed during the protocol and nothing else. While this modification is not difficult to implement in a reader phone, regular users would also need to install a modified Android library for the prover (which require root access). However, since Android is open-source, there is hope that our modification could be integrated eventually in the default library. In this case, no root access would be required for the prover.

This customization shortens the measured times as the clock runs from the HAL library layer rather than at application level. This difference is visible in our results (see Table 1) showing that the minimum measured time was reduced from 9.37 to 6.26 ms. In addition, while outliers still exist (our maximal time is close to the maximum observed in the first experiment), they become unfrequent as reflected by a median of 7.11 ms and a mean of 7.30 ms. In addition, the measurements are much more uniform, with a much smaller standard deviation of 2 ms, indicating that the values are more concentrated around the mean. If we set the error threshold E_{MAX} to be $\frac{1}{2}N_c = 16$ rounds, a choice of t_{max} within 1 ms of the minimum observed time is enough to guarantee an almost perfect correctness (as explained later).

As both types of implementations are available, we also experimented by running the application between a custom reader and a normal-mode tag using a standard Android (1 : normal ↔ custom). These results are also shown in Table 1.

Table 1. Summary of statistics from each of our implementations (the measurement values are in milliseconds).

Implementation	Min time	Max time	Standard deviation	Median	Mean
1 : normal ↔ normal	9.37	82.00	4.40	18.95	18.27
1 : custom ↔ custom	6.25	78.80	2.00	7.11	7.30
8 : custom ↔ custom	6.29	61.03	1.95	7.35	7.83
1 : normal ↔ custom	7.69	200.71	4.45	9.68	10.20

As expected the minimum value is almost the average of the 1 : normal ↔ normal and 1 : custom ↔ custom modes (7.69 ms). However, the standard deviation remains the same very large one as obtained in the 1 : normal ↔ normal case and indeed for a particular batch we did observe an outlier as large as 200.71 ms. An in-depth analysis of the outliers obtained by this implementation revealed that only around 0.1% of our measures where slower than 30 ms. We believe that these outliers must occur due to another process with more priority interrupting our protocol. This indicates that the variation in the measured time is actually caused by the prover's response times and not by the method the reader employs to measure time.

3.3 8 : custom ↔ custom

For the 8 : custom ↔ custom implementation, we modified the Swiss-Knife protocol to use 8-bit challenges and responses, but we left unchanged the *manner* of choosing the responses per each challenge bit. However, instead of processing the challenges one bit at a time and selecting the appropriate response, we rely on the smartphone's memory and computational power to *precompute* the responses to all possible challenges for each round. Since we did not modify the protocol itself, the mafia-fraud resistance for N_c fast interactive rounds and an error threshold of E_{MAX} rounds is about $2^{-8\cdot(N_c-E_{\mathsf{MAX}})} + \epsilon_{\mathsf{PRF}}$, in which ϵ_{PRF} is the advantage of a best distinguisher to tell apart the output of PRF from a truly random string. Our modification of the protocol relies on the observation that the message encoding in the ISO14443-4 standard/ISO-DEP protocol requires that the two parties exchange bytes instead of bits.

This implementation was run with 32 fast challenge/response rounds for a customized prover and verifier, yielding an approximate mafia-fraud resistance of $2^{-256} + \epsilon_{\mathsf{PRF}}$. Thus, we also need to ensure that both the output size of the PRF and the size of the secret key sk are at least 256 bits. Instead of HMAC_SHA1 we use HMAC_SHA256, which returns 32 bytes. The variables const, sk a, T^0, T^1, Vf, C, $N_{\mathcal{V}}$, and $N_{\mathcal{P}}$ are also 32 bytes long. The prover $\mathcal{P}$ precomputes for all 32 rounds the responses for each of the 2^8 possible challenges, and thus during the protocol he only performs a lookup to retrieve the precomputed result.

The results in Table 1 show that the minimal observed time and the standard deviation are almost the same as for the 1 : custom ↔ custom. However, this implementation has a much higher mafia-fraud resistance than the 1-bit version.

This opens the door to the possibility of using less rounds and larger challenges in the future. However, we are conscious that the current scope of our analysis is currently limited and we strongly encourage a thorough investigation of its consequences as future work. In the distance-bounding literature [4], there are two fundamental reasons for requiring the challenges and responses to be single-bit only. The first reason is to reduce to a minimum the processing time for the prover during fast rounds while the second one is to prevent relay attacks in which the Leech device accelerates the tag by augmenting the magnetic field it generates when querying the honest far-away prover. The first point does not hold for our implementation since we precompute all the possible responses rather than making the prover compute the response in real time. With respect to the second strategy, we did not investigate in how far an adversary can succeed by using it in our context.

4 Comparative Analysis

In this section, we compare and analyze the results obtained for the three implementations described in the previous section. First, we investigate the distributions of the measured roundtrip times, depicted in Figure 4 (left), and the density of those measurements (same figure on the right).

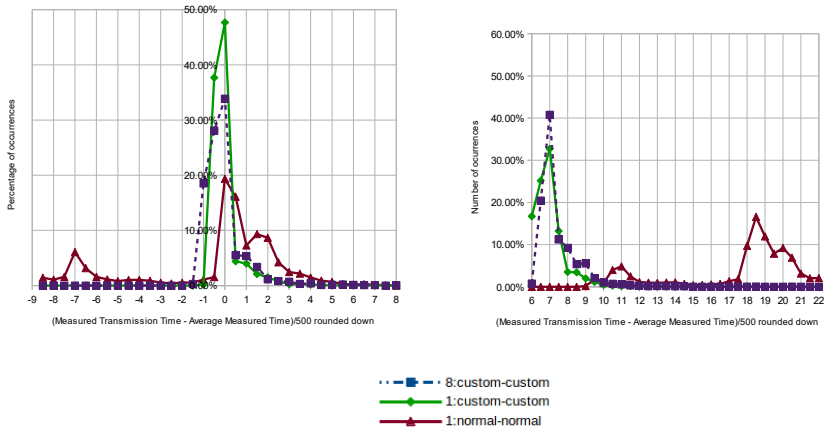

Fig. 4. Distribution of measured times around the mean (left) and density of measured times (right).

For the left plot, we first computed the mean roundtrip time $\tilde{t}$ and then the difference between each measured time and $\tilde{t}$, rounding them down to the nearest half a second. For both fully customized implementations, we observed that the

values cluster in a nearly normal distribution around the mean measured time. By contrast for the $1 : \mathsf{normal} \leftrightarrow \mathsf{normal}$ case, multiple clusters of values fall outside the neighbourhood of the mean measurement. This variation implies that even honest prover-verifier authentication can only succeed if the bound t_{max} is set to be much larger than the minimal time.

The mean measurement $\tilde{t}$ is also quite far from the minimal observed time for the $1 : \mathsf{normal} \leftrightarrow \mathsf{normal}$ distribution. This is visible in Figure 4 (right), in which the density of occurrences is shown as a percentage of the total number of measurements for each occurring value (in milliseconds). For the $1 : \mathsf{normal} \leftrightarrow \mathsf{normal}$ implementation, the minimal time is higher than for the other two implementations. Even more important, most observed times are clustered far from this minimal value. By contrast, for the $1 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ and $8 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ implementations, most values are close to the minimum. For the 8-bit version, there is a true peak in the measurements, with few values around the minimum, while for the 1-bit version the peak is less sharp, with multiple values around the observed minimum.

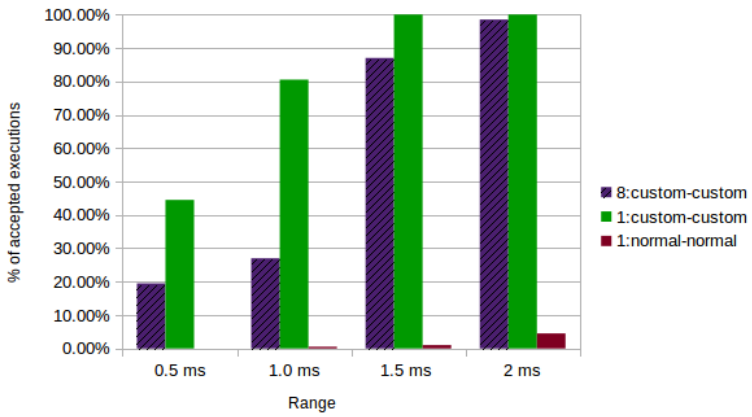

Fig. 5. Percentage of successful executions when considering $E_{\mathsf{MAX}} = 16$ rounds.

The Swiss-Knife protocol includes an error threshold E_{MAX}, such that the prover authenticates only if a number $(N_c - E_{\mathsf{MAX}})$ of challenge-response rounds are within the bound t_{max}. The higher E_{MAX} is chosen, the lower is the mafia-fraud resistance of the protocol, but the tighter one can set the bound t_{max}. We explore this relationship in more details in the following. Figure 5 displays the percentage of honest-prover/honest-verifier executions passing for $E_{\mathsf{MAX}} = 16$ rounds (and $N_c = 32$ rounds), as a function of the distance between t_{max} and the minimal observed measurement. In the $1 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ implementation, 45% of the executions succeed if t_{max} is within 0.5 ms of the minimal measured time, as opposed to only 20% for the $8 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ case. Thus,

a prover must run the $8 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ protocol on average 5 times before authenticating. The difference between implementations is much smaller if t_{max} is within 1.5 ms of the minimal time (100% for $1 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ and 87% for $8 : \mathsf{custom} \leftrightarrow \mathsf{custom}$). However, larger values of t_{max} lead to a higher risk of relay attacks.

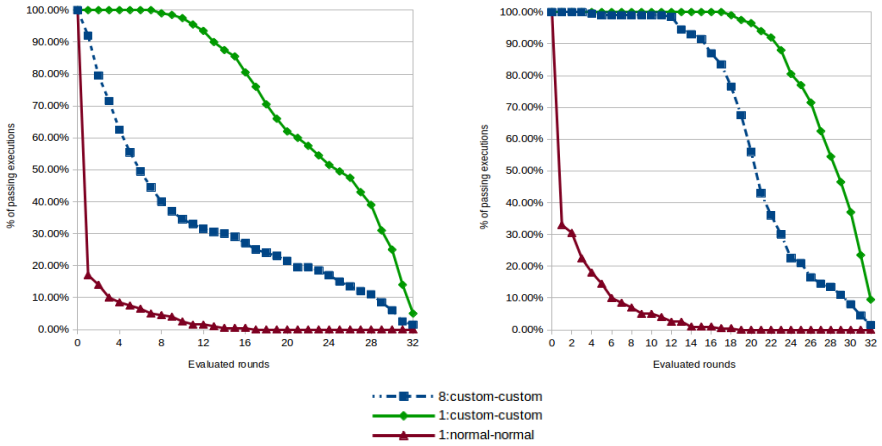

Fig. 6. Percentage of successful executions against (N_c-E_{MAX}), for t_{max} within 1 ms (left) and 1.5 ms of the minimum (right).

We also investigated the impact of E_{MAX} on the percentage of false negatives. Figure 6 shows the results obtained when t_{max} is chosen within 1 ms and 1.5 ms from the minimal measured time. The x-axis represents the number of *evaluated* rounds, which correspond to the ($N_c - E_{\mathsf{MAX}}$) rounds for which the measured time is compared to t_{max}. The $1 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ implementation always tolerates a lower value of E_{MAX} than the other versions. However, for t_{max} within 1.5 ms of the minimal measurement, the curves are closer (E_{MAX}=14 versus E_{MAX}=20), and the 8-bit protocol yields a better mafia-fraud resistance for this parameter choice. This is shown in Figure 7, in which we have 2^{-96} for $E_{\mathsf{MAX}} = 20$ for the 8-bit version versus 2^{-18} for $E_{\mathsf{MAX}} - 14$ in the 1-bit version. Thus in the long run, even with the higher E_{MAX} bound, the $8 : \mathsf{custom} \leftrightarrow \mathsf{custom}$ version tends to provide better security.

5 Future Research Directions for Distance-bounding

The results obtained by our experimental implementation of distance-bounding protocols on smartphones are highly encouraging. In particular, we demonstrate that operating the DB protocol in the HAL layer of the smartphone already

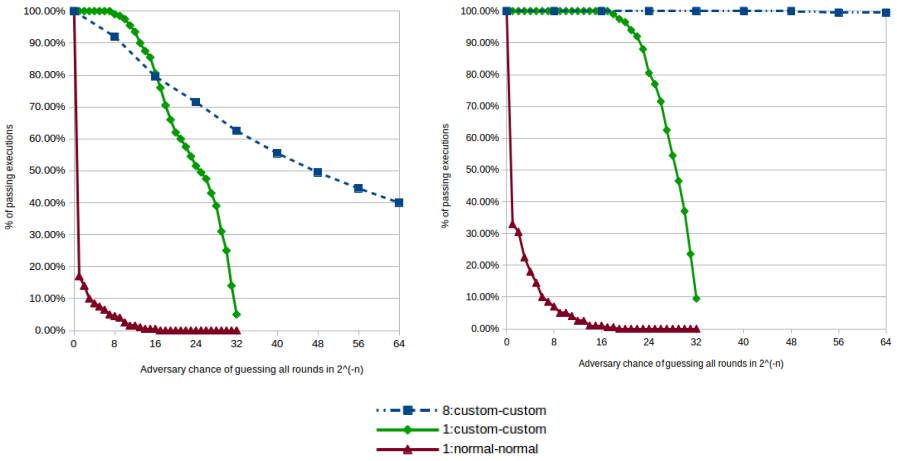

Fig. 7. Percentage of successful executions against the security level (in bits), for t_{max} within 1 ms of the minimum (left) and within 1.5 ms of the minimum (right).

provides sufficient constancy in the challenge/response time measurement for distance bounding to be effective as long as the adversary introduces a delay of more than 1.5 ms during the relay attack. While customized relay mechanisms certainly take less than this value [11], most off-the-shelf attacks seem to introduce delays of at least tens of milliseconds [8,12].

Another important conclusion of our work is that using 8-bit challenges rather than 1-bit ones increases the mafia-fraud resistance of the protocol in the long run. In this situation, one can leverage on the smartphone's memory to precompute the fast-round responses, thus lowering and stabilizing the prover's processing time. This area of research has been very little explored in the design of DB protocols thus far, thus we hope that our work will spark research towards this direction.

Another important finding of our work is that introducing a threshold for the total number of allowed errors is crucial. Indeed the variations in the measurements, even for the best executions, usually conduct to a few rounds being outside the proximity threshold. More precisely, when t_{max} is within 1.5 ms from the minimal measured time of around 6.2ms, setting values for the thresholds of $E_{\mathsf{MAX}}= 20$ and $E_{\mathsf{MAX}}= 14$, respectively for the 8-bit version and the 1-bit version, are optimal.

Overall, we believe that our three implementations have only scratched the surface of the full potential of distance-bounding on smartphones. In particular, our implementations did not rely on the SIM card in any way for the computation. Investigating this possibility would make the computational time more stable while increasing the security against side-channel attacks. In the long term, if better security guarantees could be achieved against an adversary

performing a relay attack, then the distance-bounding technology could have a fundamental impact by augmenting the payment possibilities of smartphones or providing new functionalities such as the construction of secure location proofs.

References

1. Avoine, G., Tchamkerten, A.: An efficient distance bounding RFID authentication protocol: Balancing false-acceptance rate and memory requirement. In: Proceedings of ISC'09. LNCS, vol. 5735
2. Brands, S., Chaum, D.: Distance-bounding protocols. In: Proceedings of Eurocrypt'93. LNCS, vol. 765 (1994)
3. Carluccio, D., Kasper, T., Paar, C.: Implementation details of a multi purpose ISO 14443 rfidtool. In: Printed handout of RFIDSec 06 (2006)
4. Clulow, J., Hancke, G.P., Kuhn, M.G., Moore, T.: So near and yet so far: Distance-bounding attacks in wireless networks. In: Security and Privacy in Ad-Hoc and Sensor Networks, pp. 83–97. Springer (2006)
5. Desmedt, Y., Goutier, C., Bengio, S.: Special uses and abuses of the Fiat-Shamir passport protocol. In: Proceedings of CRYPTO'87. LNCS, vol. 293
6. Fischlin, M., Onete, C.: Subtle kinks in distance bounding: an analysis of prominent protocols. In: Proceedings of WiSec'13. pp. 195–206. ACM (2013)
7. Francillon, A., Danev, B., Čapkun, S.: Relay attacks on passive keyless entry and start systems in modern cars. In: Proceedings of NDSS'11 (2011)
8. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical relay attack on contactless transactions by using NFC mobile phones. In: Proceedings of RFIDSec'10. pp. 35–49 (2010)
9. Haataja, K., Toivanen, P.: Two practical man-in-the-middle attacks on bluetooth secure simple pairing and countermeasures. Transactions on Wireless Communications 9(1), 384–392 (2010)
10. Hancke, G., Kuhn, M.: An RFID distance bounding protocol. In: Proceedings of SECURECOMM'05. pp. 67–73. IEEE Computer Society
11. Hancke G, P.: A practical relay attack on ISO 14443 proximity cards. http://www.rfidblog.org.uk/hancke-rfidrelay.pdf, accessed: 9th of january 2015
12. Henzl, M., Hanáček, P., Kačic, M.: Preventing real-world relay attacks on contactless devices. In: Proceedings of IEEE ICCST'14. pp. 376–381. IEEE (2014)
13. Hlaváč, M., Tomáč, R.: A note on the relay attacks on e-passports (2007), http://eprint.iacr.org/2007/244.pdf
14. ISO/IEC-14443: Identification cards - contactless integrated circuit(s) cards - proximity cards. Tech. rep., International Organization for Standardization (2008)
15. Juels, A.: RFID security and privacy: A research survey. Selected Areas in Communications, IEEE Journal on 24(2), 381–394 (2006)
16. Kim, C.H., Avoine, G., Koeune, F., Standaert, F.X., Pereira, O.: The Swiss-Knife RFID distance bounding protocol. In: Proceedings of ICISC'08
17. NFC Forum TM: NFC Controller Interface (NCI), version 1.1 edn. (2014)
18. NFC Forum TM: NFC Digital Protocol, version 1.1 edn. (2014)
19. Oren, Y., Wool, A.: Relay attacks on RFID-based electronic voting systems. Cryptology ePrint Archive, Report 2009/442 (2009), http://eprint.iacr.org/2009/422.pdf
20. Ranganathan, A., Tippenhauer, N.O., Singelée, D., Škorić, B., Capkun, S.: Design and implementation of a terrorist fraud resilient distance bounding system. In: Proceedings of ESORICS'12 (2012)