Chapter 7
From Relay Attacks to Distance-Bounding Protocols

Gildas Avoine, Ioana Boureanu, David Gérault, Gerhard P. Hancke, Pascal Lafourcade, and Cristina Onete

Abstract We present the concept of relay attacks, and discuss distance-bounding schemes as the main countermeasure. We give details on relaying mechanisms, we review canonical distance-bounding protocols, as well as their threat-model (i.e., covering attacks beyond relaying) stemming from the authentication dimension in distance bounding. Advanced aspects of distance-bounding security are also covered. We conclude by presenting what we consider to be the most important challenges in distance bounding.

7.1 An Introduction to Relay Attacks and Distance Bounding

In this section, we first explain the concept of relay attacks. Then, we present distance bounding, the main countermeasure, but also discuss other ways of possibly counteracting relaying.

G. Avoine (✉)
Univ Rennes, INSA Rennes, CNRS, IRISA, Rennes, France
e-mail: gildas.avoine@irisa.fr

I. Boureanu
University of Surrey, Guildford, UK

D. Gérault · P. Lafourcade
Université Clermont Auvergne, Clermont-Ferrand, France

G. P. Hancke
City University of Hong Kong, Hong Kong, PR China

C. Onete
University of Limoges, XLIM, Limoges, France

G. Avoine, J. Hernandez-Castro (eds.), *Security of Ubiquitous Computing Systems*,
https://doi.org/10.1007/978-3-030-10591-4_7

7.1.1 Relay Attacks

A relay attack against two legitimate parties A and B is one whereby a man-in-the-middle C forwards A's messages to B and/or B's messages to A, unbeknown to them. In doing so, C wishes to obtain a facility meant for A and granted by B or vice-versa. For instance, C could get to fraudulently spend the funds associated with A's bank-card at a payment terminal embodied by B.

Relay attacks are hard to detect and deter, as they subvert all conventional cryptographic mechanisms potentially employed in the protocols: C only forwards the messages, and does not need to break the cryptography that is used. This is even more acute in the case of contactless applications: user A simply brings a token (e.g., a card or phone) within range of a reader B, and the protocol starts automatically, with no consent or input by the person who is getting the privilege. Thus, a relay attack can be mounted without hindrance.

7.1.2 Distance Bounding

The further A is from B, the longer the messages relayed by C from A take to arrive at B. Hence, imposing an upper-bound on the round-trip times (RTTs) of message-exchanges was proposed as a countermeasure in [83]. This lowers the probability of successful relay attacks. This mechanism is often referred to as *distance bounding (DB)*.

The idea of distance-bounding protocols is as follows: a *verifier* (e.g., an RFID reader) is equipped in the physical layer with a reliable clock that measures the RTTs of certain communication exchanges to check that a *prover* (e.g., a card) is no further than some allowed distance. So, at some point in the protocol, the verifier starts its clock, sends a challenge, and stops the clock when it receives the response. The measured time Δ_t corresponds to twice the time it takes for a message to travel from the prover to the verifier, plus the time taken by the prover to reply. Since no information can travel faster than the speed of light c, $d = \frac{\Delta_t \cdot c}{2}$ is an upper bound on the distance between the prover and the verifier. If the prover was any further than d, then it would mean that the messages traveled faster than light, which is impossible. Consequently, if d is short enough, then the verifier can deduce that the prover is within range. In other words, a time bound $\mathscr{B}$ can be a priori fixed such that, if $\Delta_t > \mathscr{B}$, then the verifier rejects the prover.

As described above, distance bounding would be just a *proximity-checking* mechanism. However, most distance-bounding protocols do not stop at proximity-checking. Instead, they also encompass a unilateral *authentication* dimension: the prover authenticates itself to the verifier. Authentication is generally achieved cryptographically: by using well-established primitives, such as signature schemes, HMAC, encryption, and others.

7.1.3 Other Relay-Countermeasures

Approaches to relay-counteraction other than distance bounding have been proposed. In his seminal paper [178], Desmedt proposed that a prover computed his exact location on earth, signed it, and sent it to the verifier. The inconvenience in this approach is that it requires one to trust the prover not to cheat. In addition, it requires a safe localization system, which is not trivial to realize. In particular, using the GPS technology does not seem to be a robust solution [242] due to the fact that the GPS signal is sensitive to obstacles and not accurate enough. In [133], position-based cryptography is further studied and proven to be impossible.

Another option against relay attacks is to measure the strength of the signal received by the verifier [347]: since it decreases as the distance increases, it gives indications about the distance from the prover. However, an attacker can amplify the signal to make the prover appear closer to the verifier, and defeat this approach.

Similarly, a solution based on sensing the local environment (for instance the air temperature) was proposed, with the idea that if the prover was actually close to the verifier, then it would sense similar values [561]. This approach however fails if the adversary is able to manipulate the value that is being sensed, which can be relatively easy to do.

To prevent relay attacks, one can also isolate the prover inside a Faraday cage [74] during the protocol, to make sure that it cannot communicate with external entities. While efficient, this solution is not very user friendly, and severely limits the usability of the system.

Finally, radio frequency fingerprinting [496] can be used. It identifies the devices based on variations in the signal features due to imperfections in the manufacturing process. However, such fingerprinting can be counterfeited [168].

Comparing all the aforementioned relay-countermeasures, distance bounding appears the most promising option to defeat relay attacks.

7.2 Relay Attacks in Practice

Relay attacks have been implemented against contact-based smart cards [189], contactless smart cards [256], and keyless car entry systems [221]. First, in Sect. 7.2.1, we discuss attacks against “unprotected systems”. Then, in Sect. 7.2.2, taking into consideration the fact that distance-bounding type countermeasures are starting to be implemented, we consider more advanced practical relay strategies against systems thus “protected”.

7.2.1 Basic Relay Strategies

A *basic relay* equates to the attack described in Sect. 7.1.

7.2.1.1 Purpose-Built Relays

There are several relay-attack implementations against radio frequency identification (RFID) systems using purpose-built attack proxies and relay links, which incur minimal delay in executing the attack, e.g., [221, 256, 548]. The conventional approach to implementing an attack uses custom-built attack proxies, using a combination of custom hardware and hacked readers [256, 548]. The proxy will first demodulate the data symbols from the reader or token, and then forward data over an analog radio link, e.g., a video channel [256], and this tends to introduce a delay in the order of a few to tens of microseconds (2–20 μs). These implementations are also capable of active relay attacks, equivalent to a conventional man-in-the-middle or 'wedge' attack, which can modify communication with negligible additional delay, e.g., using an FPGA to reshape analog signals in real-time [256]. If the goal is to minimize the relay delay to less than a microsecond then the relay link can be implemented without demodulating the data first [221, 548]. In these cases, the proxies are either connected via a wire (120–500 ns delay), or forward data by direct up-mixing of the LF/HF carrier onto a UHF radio carrier for transmission (120–750 ns delay).

7.2.1.2 Off-the-Shelf Relays

It has also been shown that software-only implementations using off-the-shelf NFC-enabled mobile devices are effective, which simplifies the attack and allows any person with the right type of NFC-enabled mobile phone to implement a token emulator or a reader. These attacks can therefore use a standard phone as a proxy-reader and a second phone as a proxy-token and relay the data across Bluetooth, WiFi or the mobile data network [222, 392, 539]. Even though such attack implementations incur a larger attack delay (200-500 ms), these remain effective against real systems, as was demonstrated in an attack against Google Wallet [505]. There are an increasing number of non-mobile NFC devices, such as the Adafruit NFC breakout board, that easily connects with embedded hardware Arduino or Raspberry Pi, which could be used as readily available proxy platforms.

7.2.2 *Advanced Relay Strategies*

The relay attacks above were executed on systems that implemented no proximity checks. As systems start to implement such checks over conventional low-bandwidth communication channels there are practical strategies for gaining time that can hide the relay delay. Even if the attacker can gain part of a bit period, e.g., a few microseconds, it could leave enough time to mount one of the attacks in Sect. 7.2.1.

7.2.2.1 Early Send and Late Commit

If the attacker can send a challenge or response late but still get the prover or verifier to accept it as a valid message then that could also hide the relay delay. Receivers do not evaluate the bit value right at the beginning of the bit period T_B. To make the channel more reliable this evaluation is done later, in the middle or at the end of the bit period, which could be exploited to gain the attacker some time [149, 255]. For example, for NRZ (non-return to zero) coding the signal is high for the entire bit period for '1' and low for the entire bit period for '0', and the receiver samples only once in the middle of the bit period to determine the bit value, as shown in Fig. 7.1a. This means the attacker could start his response bit up to $T_A = T_B/2$ late, and still have the bit sampled correctly. Several receiver architectures, to be resistant to noise, integrate the signal across the entire bit period and evaluate the bit value at the end. In this case, the attack could 'late commit' by transmitting a larger, shorter signal later in the bit period and still achieve the same integration output at the time of bit value evaluation. If combined with early send, where the attacker guesses the value based on the observation of the first part of the bit period, the attack in Fig. 7.2 becomes possible. The attacker will guess the value of challenge C_i from the verifier early, and send it to the prover late. It will repeat this approach for the response R_i

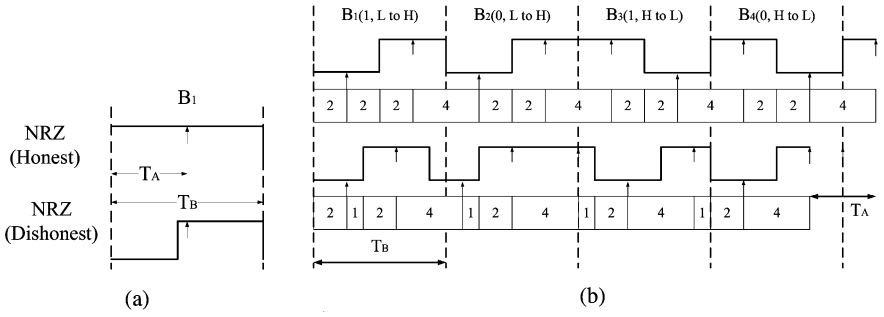

Fig. 7.1 Gaining attack time by exploiting channel characteristics. (**a**) Late commit for non-return to zero (NRZ) coding. (**b**) Speeding up Manchester code data clock [255]: Sampling clock is 8× time data clock (trigger and synchronization counter for sampling signal transition shown)

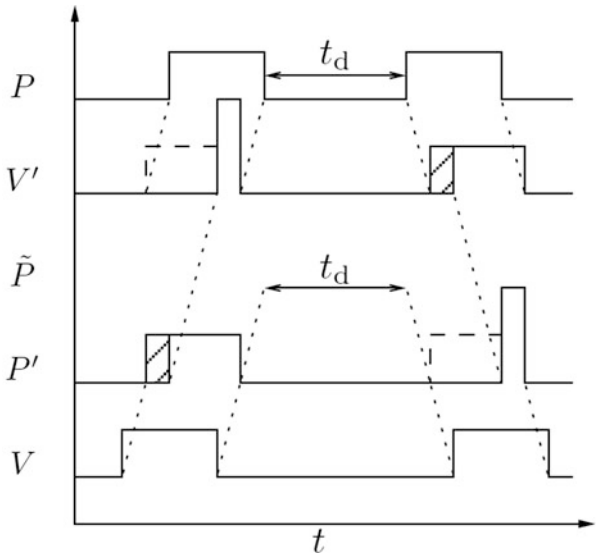

Fig. 7.2 Early send and late commit can make a Prover P appear closer than it really is. In the figure, one challenge round is relayed, with the dotted lines indicating the propagation time (the line stops and starts at transmission and reception). If the proxy-prover P' guesses C_i early, and the proxy-verifier V' commits late then the response R_i is received by the Verifier V at the same time as expected for a prover located at $\tilde{P}$ even though the Prover P is much further away [149]

that the prover sends to the verifier, and will therefore appear to be closer to the verifier than the true distance of the prover.

7.2.2.2 Speeding Up the Prover's Response

If the attacker can get the prover to provide the response earlier than expected by the verifier, then the relay delay could remain hidden, with the round-trip time of the message remaining within the bound. There are two approaches to making the prover process the challenge faster [255]. Smart tokens receive their system clock from the reader, with contact-based cards having a clock line and contactless cards recovering a clock from the received radio carrier. This allows the proxy-verifier to overclock the token, which causes the response to be calculated and transmitted earlier. If the token has its own, independent clock, then the attacker can also gain some time by exploiting data clock recovery from the data coding. For example, for Manchester coding ('1' is high to low,'0' is low to high) each bit period has an edge transition to which the receiver can synchronize its decoding data clock. If the transition is moved slightly ahead in each bit, as shown in Fig. 7.1b, then the receiver will sample earlier as the message is received and the message is decoded T_A faster than normal. This approach can also effect distance fraud if a dishonest prover is able to speed up its own response, either by calculating a

response faster than expected or sending a correct response early, e.g., if the verifier sends a challenge followed by some framing bits and expects the provers to start calculating the response only once the entire message, including the stop frame bit, is received but instead the prover can send the correct response immediately after the challenge bit is received.

7.3 Canonical Distance-Bounding Protocols

In this section, we describe and discuss two protocols that can be considered the cornerstones of distance-bounding schemes. The Brands-Chaum protocol is the earliest distance-bounding protocol ever published, and is based on Beth and Desmedt's [83] idea that roundtrip times (RTTs) can detect mafia fraud. The Hancke-Kuhn protocol resurrected research interest in distance-bounding protocols, and was specifically designed for contactless devices.

7.3.1 General Structure

General Setup Distance-bounding schemes can use either symmetric- or public-key cryptography. In the symmetric-key scenario, the prover and verifier share a secret key K. For public-key primitives, the prover stores a private/public key-pair (sk_i, pk_i), for which the verifier only holds the public key. Each verifier is assumed to possess a clock able to measure roundtrip times (RTTs) with a fine-grained resolution (ideally, less than a nanosecond). In the protocol, the verifier uses the clock to measure RTT values for several so-called *time-critical* rounds.

General 3-Phase Structure The general structure of distance-bounding protocols follows these three phases (each consisting of zero, one, or multiple rounds of communication): session set-up, proximity checking, and verification. During *session set-up*, the prover and verifier exchange session-specific data and possibly pre-compute some values that will be used during the next stage. During *proximity checking*, the parties execute n fast phases of communication: the verifier generally starts the clock at the beginning of each round, and stops it at the end. The responses r_i sent by the prover, and the round-trip time (RTT) of each round are stored by the verifier. Finally, during *verification*, the verifier performs some cryptographic operations, may exchange some more messages with the prover, and it compares the measured RTT values of the proximity-checking phase with a threshold. At the end of this phase, the verifier must output an *authentication bit*, which is typically 1 if the prover is assumed to be legitimate and within a correct distance, and 0 otherwise.

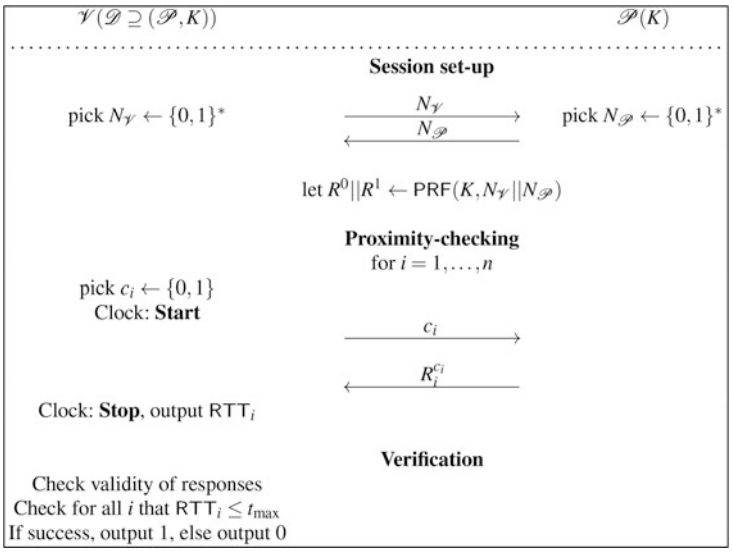

Fig. 7.3 The Hancke and Kuhn protocol between a prover $\mathscr{P}$ and a verifier $\mathscr{V}$. The notation || describes string concatenation

7.3.2 The Hancke-Kuhn Protocol

The protocol presented by Hancke and Kuhn [254] in 2005 performs symmetric-key distance bounding. It relies on a pseudorandom function (PRF), which takes two inputs, a key and a message, and outputs a string of fixed length (in our case, $2n$). Figure 7.3 depicts this protocol for a prover $\mathscr{P}$ and a verifier $\mathscr{V}$. At *session set-up*, $\mathscr{P}$ and $\mathscr{V}$ exchange nonces.[1] The two parties then use the PRF to map the key K and the concatenation of the two nonces to a bit-string of length $2 \cdot n$. This value is divided into a left and a right register of length n each, which we denote R^0 and R^1 respectively. During *proximity-checking*, in each of the n subsequent fast rounds, the challenger chooses a bit c_i at random, and the prover is expected to respond with the i-th bit from either the left response register (if $c_i = 0$) or from the right one. We denote these bits R_i^0 and R_i^1 respectively. For each round, the RTT is measured. At the end of the protocol, during *verification*, the prover is authenticated if, and only if, all the responses provided by the prover were correct, and if all the measured RTT values are under the $t_{\max}$ bound.

Design Intuition As long as the key K is unknown to an attacker, the PRF guarantees the security-crux herein: two independent response strings. Indeed, a man-in-the-middle attacker can relay the exact nonces used by an honest prover and

[1] In an early version of this protocol, only $\mathscr{V}$ sent a nonce; $\mathscr{P}$ did not. That version of the protocol is insecure against worst-case attackers; thus we choose to present a later version here.

verifier. This allows the adversary to establish two sessions (one with the prover, the other, with the verifier) which share the same response strings R^0 and R^1. This adversary can now use its session with the prover to extract data: before it receives the honest verifier's challenge, the adversary can query the prover with any kind of request. If the protocol were to rely on only one response string, the adversary could obtain the entire response and forward it to the attacker.

7.3.3 The Brands-Chaum Protocol

The public-key counterpart of the Hancke-Kuhn protocol was proposed by Brands and Chaum [113] and relies on commitment schemes and digital signatures. Commitment schemes allow users to temporarily hide a value; the commitment will also only open to that hidden value, and not to any other. Signature schemes are public-key primitives allowing a signer to generate signatures for a given message and a secret key; the signature can be verified for that message with the public key.

Figure 7.4 depicts an execution of the Brands-Chaum protocol. The session set-up and verification consist of one-message rounds each. During *set-up*, the prover chooses and commits (in a message C) to a number of responses to be used at proximity checking. Note that C *hides* the contents of the message, from both an attacker and the verifier. In each round of the *proximity-checking phase* the verifier picks a one-bit random challenge c_i and sends it to $\mathscr{P}$. The latter's response is $c_i \oplus r_i$, where r_i is the response bit to which the prover committed for this round. The values R_i and the measured RTT values are stored by the verifier. Finally, during *verification*, $\mathscr{P}$ sends to $\mathscr{V}$ the opening of the commitment C and a signature on the concatenated challenge and response values exchanged at proximity-checking. The

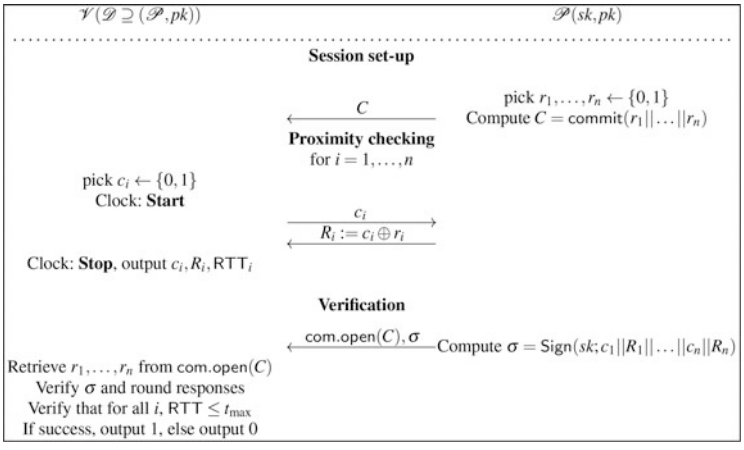

Fig. 7.4 The Brands–Chaum protocol for a prover $\mathscr{P}$ and a verifier $\mathscr{V}$

verifier retrieves the randomly-chosen r_i values from C and uses them to ascertain the validity of the prover's time-critical responses and the signature σ and R_i values. If these values verify and the measured RTTs are below the $t_{\max}$ bound, then the verifier authenticates the prover.

Design Intuition The commitment serves a dual purpose: it hides the values of r_i until they become useless to the attacker (i.e., until after the proximity-checking rounds); and the commitment compensates for the fact that the response values are chosen entirely by the prover. Finally, the commitment allows the verifier to retrieve the r_i values without exchanging or sharing any further keys with the prover. The commitment, however, does not authenticate $\mathscr{P}$; that is achieved by the signature σ. The signature also effectively prevents *pre-ask* strategies during the proximity-checking phase.

7.4 Distance-Bounding Threat Model and Its Formal Treatments

In this section, we present the main threats in distance bounding, and the state of formal security analysis in this field. We also review more recent protocols, comparing their advantages and disadvantages.

7.4.1 Main Threat-Model

Distance-bounding schemes are vulnerable to attacks other than relaying, issued out of the proximity-checking measure. For instance, any attack that makes the prover appear closer than it actually is defeats the purpose of a distance-bounding protocol, which is to compute a correct upper bound on this distance. The threats we present can be classified as attacks by outsiders and attacks by insiders. In the first category lies mafia fraud, where an unauthorized adversary attempts to be accepted by the verifier. In the second, comprising distance fraud, distance hijacking and terrorist fraud, a faraway dishonest prover attempts to be accepted by the verifier despite his distance.

7.4.1.1 Mafia Fraud (MF) [178]

In mafia fraud, an adversary $\mathscr{A}$ authenticates in the presence of a far-away honest prover. A mafia fraud typically involves a faraway prover, and two collaborating adversaries: one near the prover, and one near the verifier. The fraud succeeds if the authentication of the adversary located close to the verifier is accepted by the verifier.

7.4.1.2 Distance Fraud (DF) [113]

In distance fraud, a malicious prover located far away from the verifier attempts to convince the verifier that he is close. The fraud succeeds if the authentication of the faraway malicious prover is accepted.

7.4.1.3 Distance Hijacking (DH) [160]

Distance hijacking is a distance fraud in which honest provers are present near the verifier. This gives the malicious prover more surface of attack, so that some protocols are resistant to distance fraud, while being vulnerable to distance hijacking. For instance, in the Brands-Chaum protocol (Sect. 7.3.3), which is resistant to distance fraud, a faraway prover can eavesdrop on a session played by an honest prover P (located close to the verifier), send the final message before P does, and be authenticated in place of P. Distance hijacking succeeds if the verifier accepts the authentication of the faraway malicious prover.

7.4.1.4 Terrorist Fraud (TF) [178]

Terrorist fraud is an attack in which a malicious prover, located far away from the verifier, is helped by an accomplice located near the verifier. A trivial attack in this scenario would be that the prover simply gives all his secret keys to his accomplice. Since this attack cannot be prevented if the prover has access to his secret key, we make the additional assumption that the prover does not want the accomplice to impersonate him later. Hence, a terrorist fraud succeeds if the verifier accepts the authentication of the faraway prover through his accomplice, and the accomplice cannot authenticate on his own in a later execution of the protocol.

7.4.2 Provable Security and Formal Verification

Provable security is the field of research which aims at building formal, mathematical proofs of the security of systems or protocols. Early distance-bounding protocols were analyzed in an ad hoc fashion, so a call for provable security of distance bounding was needed. A preliminary framework [41] for modelling distance bounding paved the way to formal treatment of distance-bounding security.

In terms of formal security for distance bounding, we have: computational formalisms [110, 193], and symbolic ones [177, 401]. Computational models treat the messages as bitstrings and attackers as probabilistic polynomial-time algorithms trying to defeat cryptographic goals. Symbolic security verification represents messages as terms in a term algebra, abstracts the cryptographic primitives to black box functions and models attackers as rules manipulating the terms and black-box

cryptographic functions. Due to these abstractions, symbolic models are easier to mechanize into automatic verifiers, yet generally an attack found in such models is more a logical flaw than a cryptographic-design problem.

7.4.2.1 Symbolic Verification

The two symbolic models permit us to use semiautomatic tools, Tamarin [406] and Proverif [93], respectively, to verify the security of distance-bounding protocols. They slightly differ in their approach: [177] models time and distance explicitly, while [401] abstracts this into some classification of the order of messages. However, they find similar attacks. Moreover, both methodologies take a step beyond the scope of previous computational models: they consider that the verifiers can be corrupted. Also, outside formalizations, in distance bounding, verifiers were traditionally considered honest (except when user privacy is considered).

However, as symbolic models, there are some attacks that they cannot find, due to the abstractions they make. For instance, if a prover is within the distance bound, it might be possible for a mafia-fraud adversary to flip challenge bits on the fly without being detected, which allows him to recover the secret key of the provers in some protocols [62]. This kind of attack can be found using the computational models, but not the symbolic ones, which abstract bitstrings to terms.

7.4.2.2 Provable Security

Due to abstracting the cryptographic primitives into black-boxes, symbolic-verification mechanisms also cannot detect attacks by "PRF programming" [108]. Some protocols, such as Swiss-Knife or Hancke-Kuhn, use a PRF to compute the response vectors. However, as noted in [108], the pseudorandomness of a PRF is only guaranteed if the adversary does not know anything about the involved key and if there is no oracle/reuse for/of the key anywhere else in the protocol. Yet dishonest provers in distance-fraud attacks do know the key of the PRF. And, in distance-bounding protocols such as the Swiss-Knife protocol [328], the key is re-used outside of the PRF call in forming the responses. So, [108] exhibit "programmed PRFs": dishonest provers can use the PRF to mount distance fraud, and man-in-the-middle attackers can adaptively chose inputs to mount mafia fraud. In turn, this means that in provably-secure distance bounding, care needs to be taken with security claims resting just on pseudorandomness.

For both symbolic and computational models, modelling terrorist fraud is a big challenge. The symbolic models for terrorist fraud are either too strong or too weak, and the computational ones are often tailored definitions proposed for specific protocols. For instance, SimTF [216] imposes restrictions on the communications between the prover and his accomplice, and in [109], the prover helps his accomplice several times instead of just once.

7.4.2.3 Provably-(in)Secure Protocols

Designing a distance-bounding protocol that is both efficient and provably-secure has proved a difficult task.

For instance, the Hancke-Kuhn scheme presented in Sect. 7.3 only provides sub-optimal mafia-fraud resistance (3/4 per round as opposed to the optimal 1/2); in addition, it is vulnerable to distance frauds by PRF-programming. Striving for optimal mafia- and distance-fraud resistance, Avoine and Tchamkerten [45] describe a scheme in which the proximity-check responses are inter-dependent: this strategy makes the per-round mafia-fraud security asymptotically approach the optimal bound of 1/2, but fails to thwart PRF-programming strategies. By combining a late authentication like Brands-Chaum and two pseudorandom response registers like Hancke-Kuhn, Kim et al. attempted to achieve optimal mafia- and distance-fraud resistance, as well as terrorist-fraud resistance [328]. However, its design includes a circularity in the use of the key which does not allow provable mafia-fraud resistance; in addition, its use of PRFs is problematic with respect to achieving distance-fraud resistance.

Protocols that provably guarantee the four properties described above are rare in the literature [40, 114]. The SKI protocols [110] introduced a new countermeasure to terrorist fraud by using a leakage function. This design is further refined and made efficient by Boureanu and Vaudenay [111, 325]. A recent protocol called SPADE [118] circumvents PRF-programming attacks by using one-time keys during the proximity-checking phase; in that case, terrorist-fraud resistance is achieved by adding in a backdoor. An extended family of protocols using the same basic designs was proposed in [42]; it can be instantiated with various primitives, achieving different degrees of provable security and privacy.

The reader is referred to extensive distance-bounding surveys, such as [40, 114].

7.5 Distance-Bounding Protocols in Practice

7.5.1 NXP's Mifare Technology

NXP is a world-wide semiconductor supplier especially involved in secure identification, automotive and digital networking industries. Mifare is a series of NXP's contactless products that includes four families, namely Classic, Plus, Ultralight, and DESFire. Mifare Plus (X and EV1) as well as Mifare DESFire (EV2) benefit from a distance-bounding protocol [445, 446]. Note that the DB protocols are not activated by default on these cards, and the data sheets do not explain how the system operator should evaluate the value of the round-trip time upper bound.

Although the protocols have not been publicly released, it is worth noting that NXP published several patents on distance-bounding protocols. Figure 7.5 depicts the protocol described in [303, 553]. In contrast to most DB protocols available in

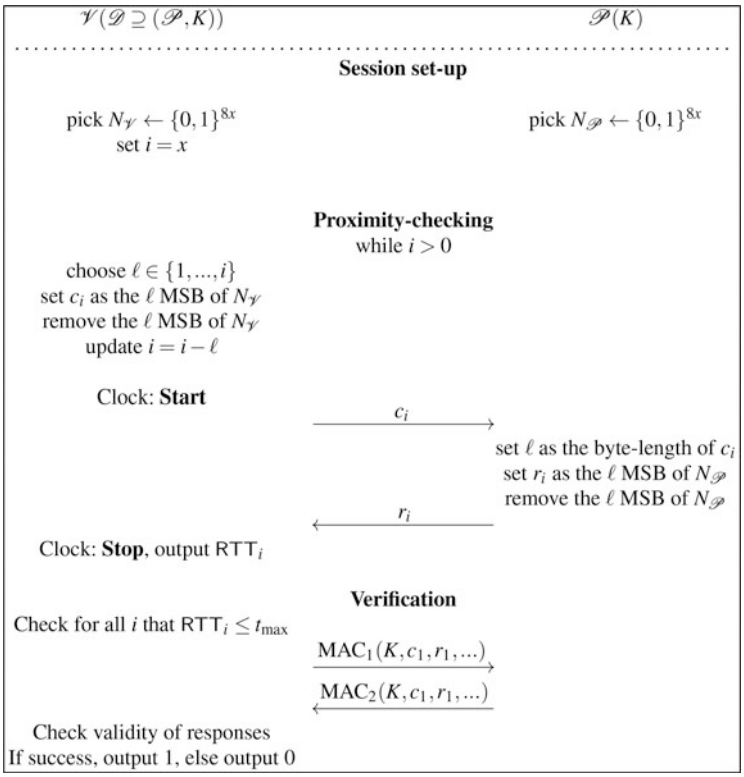

Fig. 7.5 Sketch of the patented NXP DB Protocol [303, 553]

the literature, this NXP DB protocol is byte-oriented, meaning that the messages of the fast phase contain one or several bytes instead of a single bit. The byte-length x of the random values is not enforced in the patents, but suggested only. They can typically be 7 or 8 bytes. The fast phase is followed by a verification phase where MACs are exchanged. The MACs are computed "over the complete 7-byte random numbers and some information about the speed at which the [reader] and [transponder] operate". Note that "the random number ordering for the MAC input reflects the same split as during the sending of the proximity check commands." Obviously, the two final MACs must contain the message direction to avoid a trivial reflection attack. The NXP DB protocol is unlikely to be resistant to purpose-built relays—because the measurement resolution is probably not high enough to detect fast relays—but it might resist off-the-shelf relays.

7.5.2 3DB Technology

3DB Access AG is a Swiss company founded in 2013, by Boris Danev and David Barras. 3DB developed an integrated circuit that contains a distance-bounding protocol based on Ultra-Wide Band (UWB) pulses compliant to IEEE 802.15.4f. The technology allows a reader to estimate the distance to reach a given contactless receiver. It aims to avoid mafia-fraud attacks, but it does not consider the other frauds presented in Sect. 7.4 (e.g., it does not consider distance fraud). The distance range is 120 meters (line of sight) and the accuracy of the distance-bounding protocol is 10 cm according to the product's datasheet.[2] The 3DB technology specifically (but not only) targets the market of keyless entry and start systems (PKES), given that such systems are particularly vulnerable to relay attacks [221]. It is likely that most vehicles will be equipped with such a DB-friendly PKES in the future.

The protocol implemented in the 3DB technology, described in [531], is based on the Brands-Chaum protocol [113]. However, it takes the channel characteristics into account and includes countermeasures to thwart physical-layer attacks, in particular the "early detect and late commit" attack described in Sect. 7.2 that is mitigated since the basic symbol pulses have a very short period. These countermeasures rely on the reordering and blinding of the pulses. The reordering consists in applying a permutation to the pulse positions associated with each bit. The number of bits considered in the pulse reordering is actually an adjustable security parameter. The blinding consists in XORing the stream of pulses with a mask. The cryptographic primitives used to generate the permutation and the mask are not described. No attack has been suggested so far on these reordering and blinding techniques. Apart from security properties, UWB channels can also provide very accurate time-of-arrival measurement as the timing resolution achievable with a signal of bandwidth B, is $1/2B$.

7.5.3 Relay-Resistance in EMV

Relay attacks are particularly relevant in contactless-payment systems. Indeed, no PIN code or other payee-originating input is requested with such payments. Moreover, most contactless payment cards rely on ISO 14443, which is a standard available in most of today's smartphones. Consequently, performing a relay attack between a payment terminal and a payment card is as simple as uploading an app on a smartphone [567].

Indeed, using off-the-shelf smartphones and some in-house Android software, this relay threat was exhibited in practice by Chothia et al. [141] against the EMV (Europay, Mastercard and Visa) contactless-payment protocol; this is the most wide-

[2] Available at the 3DB Access AG Website, https://www.3db-access.com/, May 2018.

spread type for contactless payments. In their work, Chothia et al. also introduced a countermeasure to mitigate their own relay attack. Their so-called PaySafe protocol is put forward as a slight variant of the contactless version of PayWave, i.e., the EMV protocol used by Visa. In PaySafe, a new command is introduced into the EMV contactless protocol such that a calculation of the round trip times becomes possible for EMV readers. Namely, the reader sends a nonce to the card and expects that the latter will respond with a pre-generated nonce; the reader measures the time taken by the whole exchange and if it is beyond a pre-established bound, then the reader aborts the protocol. In PaySafe, the nonces used in this timed phase are encompassed in some other messages, included in a MAC issued by the card and keyed on a key the card shares only with the bank.

It is worth noting that PaySafe did not aim to be a full distance-bounding protocol (i.e., it did not mean to protect against the distance-bounding frauds presented in Sect. 7.4)

EMVCo—is the consortium behind EMV—give the EMV contactless payments' specifications in [199] (current version is 2.7, April 2018). Since 2016, these specifications include the possibility for a *relay-resistance mechanism*, which is inspired by PaySafe [141]. A friendly introduction to this protocol is provided in [563]. As of today, there are unfortunately no public figures about the number of MasterCard/Visa readers that benefit from this feature.

7.6 Current Challenges in Distance Bounding

7.6.1 Theory vs. Practice

Provable-security/formal-methods models for DB (see Sect. 7.4) generally do not capture accurately the DB threats shown in practice. For instance, one major assumption that most DB formal models make is that the computation on the prover's side, during the timed exchanges, is instantaneous or constant. In practice, as [141] showed, different cards have significantly distinct response-times, leading to practical attacks which cannot be easily found via theoretical tools.

Besides such coarse abstractions, other approximations are made by provable-security models for cryptographic-proofs to become possible. For instance, in some variants of the model in [193], no communication is allowed between colluding attacking parties during the timed phase (i.e., the coalition has to be active outside the timed phase). Or, in the formalism in [110], the time taken to compute over bits equal to 0 is always considered the same as that to compute over bits equal to 1, which—as Sect. 7.2 explained—is not always factually true. These two approximations entail that the respective models are too weak. But also there is the possibility that some formal security definition is too strong, i.e., that it would classify a protocol as insecure when in practice the protocol is secure (see [216]).

Last but not least, the theoretical DB protocols presented in Sect. 7.3 follow a design whereby the fast phase is generally formed of a repetition of a number of timed rounds, where each challenge/response is one bit. These designs (endorsed by formal models/proofs, etc.) were traditionally anchored in practice, and Sect. 7.2 alluded to this: i.e., a challenge given as a bitstring can lead to bit-by-bit early reads and therefore possible early responses by dishonest provers. But, *as of recently*, there seem to be mechanisms for these early-send attacks to be effectively counteracted by other ingenious, practical mechanisms in designs even in cases where the timed challenges/responses are bitstrings (see Sect. 7.5 or [531]). However, it is important to recall that the security of the DB design in [531] has not yet been formally analyzed, and the protocol only claims to protect against relay attacks, not other DB threats.

7.6.2 Application-Aware DB

In the formal models presented in Sect. 7.4 and even in the practical considerations given in Sect. 7.2, we saw that the DB threat-model has thus far been generally focused on this primitive in isolation; that is, it assumes an honest verifier, a dishonest prover and a malicious man-in-the-middle. However, as DB is adopted in different applications (e.g., PKES as per the above), these security considerations will need adjustments. To begin with, the verifier may be dishonest, or some threats—such as terrorist fraud—may become irrelevant, or specific anonymity concerns may be considered. In this space of fine-tuned threat models for DB, two lines have recently emerged [107, 326]. Namely, [107] advances a formal DB threat-model where a fine-grained level of corruption of the prover (i.e., white-box, black-box) is taken into account, such that each application can "pick and choose". In turn, this also leads to clear-cut, DB-security properties and even the exclusion of resistance to terrorist fraud, in some cases. Complementary to this, [326] recently advances a formal DB model with three parties, where the new party is a named piece of hardware and this also leads to a fine taxonomy of DB-security properties, with an application-ready nature.

DB efficiency is paramount, but it varies from application to application. A DB solution that can be acceptable on a smartphone, may be unacceptable on a simple, passive card. A series of research lines [111, 325] discussed the efficiency of DB protocols with "traditional" structure, i.e., following the designs presented in Sect. 7.3, from a theoretical-analysis viewpoint. At the same time, the practical solution for proximity-checking in PKES offered by 3DB (see Sect. 7.5) is extremely efficient in practice. However, this question of efficiency stands, especially if new DB solutions are to be given on top of different applications, such as EMV.

In DB adoption, there are also strong backwards-compatibility constraints. For instance, in EMV, the public-key infrastructure or the restrictions of keeping as

close as possible to old-generation EMV cards/readers are such that a DB protocol, following the designs we saw in Sect. 7.3, is simply un-adoptable out of the box.

7.6.3 *Specialist Implementations and Slow Adoption*

On the one hand, PKES with relay-protection are finally becoming commercial—arguably due to relay attacks being exploited by fraudsters in the automotive sector. On the other hand, in DB-enhanced EMV contactless protocols (*à la* PaySafe), a dishonest party already has a tangible incentive to mount a distance-fraud attack;—a purchase receipt carries an intrinsic proof that the card was in the range of the reader. Yet, EMV with relay-protection is not widely deployed and, indeed, the markets do not appear to call for protocols to be enhanced with full DB-protection yet.

Should such DB frauds appear in practice, would we then see fully-fledged DB solutions being implemented for commercial purposes? Or, will the 5th generation of mobile networks (5G) and its increased spectrum and higher bands lead to the true rise of DB technology in the ubiquitous systems of the 2020s, and raise new DB research questions?