Distance Hijacking Attacks on Distance Bounding Protocols*

Cas Cremers[1], Kasper B. Rasmussen[2],
Benedikt Schmidt[1], and Srdjan Čapkun[1]

[1]ETH Zurich, Institute of Information Security, Switzerland.
[2]University of California, Irvine, Computer Science Dept., California.

Version 3.0, 28 August 2012

Abstract

After several years of theoretical research on distance bounding protocols, the first implementations of such protocols have recently started to appear. These protocols are typically analyzed with respect to three types of attacks, which are historically known as Distance Fraud, Mafia Fraud, and Terrorist Fraud.

We define and analyze a fourth main type of attack on distance bounding protocols, called Distance Hijacking. This type of attack poses a serious threat in many practical scenarios. We show that many proposed distance bounding protocols are vulnerable to Distance Hijacking, and we propose solutions to make these protocols resilient to this type of attack.

We show that verifying distance bounding protocols using existing informal and formal frameworks does not guarantee the absence of Distance Hijacking attacks. We extend a formal framework for reasoning about distance bounding protocols to include overshadowing attacks. We use the resulting framework to prove the absence of all of the found attacks for protocols to which our countermeasures have been applied.

Previous proposals for distance bounding protocols only analysed their protocols with respect to some specific attack types, whose relations and problem coverage are unknown. To improve this situation, we define an exhaustive classification for attacks on distance bounding protocols.

Keywords: Distance bounding, location verification, position verification, attacks, hijacking, multi-prover environment, formal model, formal verification

1 Introduction

By using *distance bounding protocols*, a device (the verifier) can securely obtain an upper bound on its distance to another device (the prover). A number of distance bounding protocols were proposed in recent years [2, 4, 5, 12, 14, 15, 19–21, 23, 24,

*An extended abstract of this paper appears at IEEE S&P 2012.

27–29]. The proposed protocols differ in terms of the performance and security guarantees that they provide. So far, several distance-bounding protocols were implemented, some using digital processing and short symbols [9, 16], whereas others rely on analog processing and use signal streams (operating similarly to radar systems) [23].

The security of distance-bounding protocols was so far mainly evaluated by analyzing their resilience to three types of attacks. For historical reasons, these are known as *Distance Fraud*, *Mafia Fraud* and *Terrorist Fraud*. In Distance Fraud attacks, a sole dishonest prover convinces the verifier that he is at a different distance than he really is. In Mafia Fraud attacks, the prover is honest, but an attacker tries to modify the distance that the verifier establishes by interfering with their communication. In Terrorist Fraud attacks, the dishonest prover colludes with another attacker that is closer to the verifier, to convince the verifier of a wrong distance to the prover. So far, it was assumed that distance bounding protocols that are resilient against these three attack types can be considered secure.

However, we show that many of these protocols, irrespective of their physical-layer implementation, and including the classical Brands and Chaum protocol [4] and the recent CRCS protocol [23], are vulnerable to attacks when used in environments with multiple provers. We coin this type of attack *Distance Hijacking*. In Distance Hijacking attacks, a dishonest prover convinces the verifier that it is at a different distance than it actually is, by exploiting the presence of an honest prover. For example, one of the ways in which the dishonest prover can achieve this is by hijacking the distance measurement phase of a distance bounding protocol from an honest (closer or further) prover. This type of attack can pose a serious threat in many practical scenarios.

Conceptually, Distance Hijacking can be placed between Distance Fraud and Terrorist Fraud. Unlike Terrorist Fraud, in which a dishonest prover colludes with another attacker, Distance Hijacking involves a dishonest prover interacting with other honest provers. Unlike Distance Fraud attacks, which involve only a dishonest prover and a verifier, Distance Hijacking attacks additionally involve other honest provers. These differences have significant consequences. For example, the countermeasures proposed against Terrorist Fraud rely on the assumption that dishonest provers are not willing to share their keying material with other attackers. Such countermeasures will therefore not deter the dishonest provers from executing Distance Hijacking attacks that do not involve other attackers. Furthermore, Distance Hijacking can occur even in situations where Terrorist Fraud is not a concern. In fact, as we will show, protocols that are resilient against the three classical attack types may still be vulnerable to Distance Hijacking.

We define an exhaustive classification for attacks on distance bounding protocols that includes Distance Hijacking. Our classification naturally leads to minor reformulations of previously known attack types. Instead of using the traditional attack names for our new definitions, we propose names that are more descriptive and less generic.

We perform a case study of existing protocols. All distance bounding protocols

that were proposed in the last years roughly fall into two categories: those based on the Brands and Chaum protocol, and those based on the Hancke and Kuhn protocol. We show that all proposed protocols that followed the structure proposed by Brands and Chaum are vulnerable to Distance Hijacking. Protocols that followed the structure proposed by Hancke and Kuhn are less vulnerable to this type of attack. We propose two classes of effective and generic countermeasures that make Brands and Chaum and related protocols secure against Distance Hijacking in the above scenario. Our countermeasures are inexpensive: the protocols can be repaired without introducing additional messages or cryptographic operations.

Remarkably, none of the existing frameworks for analyzing distance bounding protocols (e. g., [1,3,10,18,25]) guarantees the absence of our Distance Hijacking attacks, even if some instances of Distance Hijacking can be detected using some of those frameworks. We extend the formal framework of Basin et al. [3] to capture all known types of Distance Hijacking attacks and use the resulting framework to analyze several protocols. The new framework enables us to model bit-level manipulations of messages by considering overshadowing parts of a message [22], as well as flipping some bits of a message. We use our framework to formally prove for specific protocols that our fixes indeed prevent the found attacks.

We show that all distance bounding protocols, including those based on the Hancke and Kuhn protocol, may be vulnerable to Distance Hijacking if run alongside another distance bounding protocol. This can occur if more than one distance bounding protocol is used in the same environment, i. e., a multi-protocol environment. In particular, some protocols, when run by an honest prover, enable a dishonest prover (running, e. g., a Hancke and Kuhn protocol) to hijack the distance of the honest prover. Such attacks can be seen as a variant of the Chosen Protocol Attack [13]. However, unlike Chosen Protocol attacks, our attacks do not require the protocols to share any cryptographic material. We discuss designs of distance bounding protocols that enable such attacks and show how to mitigate these attacks.

Contributions First, we identify Distance Hijacking as a threat for distance bounding protocols that are run in multi-prover environments, whose absence is not guaranteed by existing frameworks. Second, we show that prominent distance bounding protocols are vulnerable to Distance Hijacking and propose countermeasures. Third, we extend a formal framework for reasoning about Distance Bounding protocols to model overshadowing attacks and use the resulting framework to prove correctness of our countermeasures for specific protocols. Fourth, we address the security of distance bounding protocols in multi-protocol environments and propose mitigating measures. Finally, we generalize Distance Hijacking to Location Hijacking, and show that it is possible to hijack locations at which no other provers reside.

We proceed as follows. In Section 2 we provide background on distance bounding protocols. In Section 3 we introduce Distance Hijacking attacks and analyze

the resilience of existing distance bounding protocols against these attacks. We relate the attacks to the classical attack types and provide an exhaustive classification. In Section 4 we show how distance bounding protocols can be made resilient against Distance Hijacking. In Section 5 we present an extended formal framework and analyze a set of protocols. In Section 6 we analyze the resilience of distance bounding protocols to Distance Hijacking in multi-protocol environments. We introduce the notion of Location Hijacking in Section 7, present the related work in Section 8, and conclude in Section 9. In Appendix A we provide a detailed attack on the signature-based version of the Brands and Chaum protocol.

2 Background

The goal of a distance bounding protocol is to enable a verifier to establish an upper bound on its physical distance to a prover. As a running example, we consider the basic Brands and Chaum protocol with signatures [4, p. 7], depicted in Figure 1. In the protocol, the prover P randomly generates (denoted by $\in_R$) a bit string $m_1, \ldots, m_k$, and sends a commit of this value to the verifier V. Thus, although the bit string is not revealed yet, V will be able to check whether P indeed committed to this particular string when V learns the string later. The verifier then generates his own random bit string $\alpha_1, \ldots, \alpha_k$, and initiates the so-called *rapid bit exchange*. In this exchange, bits are sent one-by-one, and the prover has to respond as quickly as possible with the exclusive-or ($\oplus$) of the challenge bit string α and his own bit string. In the end, the verifier will derive an upper bound on the distance to the prover from the response times. Notice that the prover can delay messages at will, making himself appear farther away, but he cannot respond faster than what is dictated by the time-of-flight of the messages. After this phase, P concatenates the bits as c, and sends to V a means to open the commit he sent earlier, as well as the concatenation c signed with his signature key. Upon receiving this final message, V verifies that the commit previously sent by P indeed matches with the response (by computing $m_i = \alpha_i \oplus \beta_i$ and opening the commit), concatenates the bits he has observed, and compares them to the received signature using the public key of P.

Because the goal of a distance bounding protocol is to provide a guarantee for the verifier V, V will never participate in an attack since that would mean V would be attacking itself. The attacker can of course pretend to be another verifier V', and abuse his location to attack the real verifier V.

As stated in the introduction, three different classes of attacks are traditionally considered in the analysis of distance bounding protocols: Distance Fraud, Mafia Fraud and Terrorist Fraud. All attacks that fall into one of these three classes have a similar goal, namely to make the verifier believe that the prover P is physically closer to the verifier V than it really is. The main difference between these attacks is in the parties that carry out the attack, and their mutual relationships.

Mafia Fraud attacks, also called relay attacks, were first described by

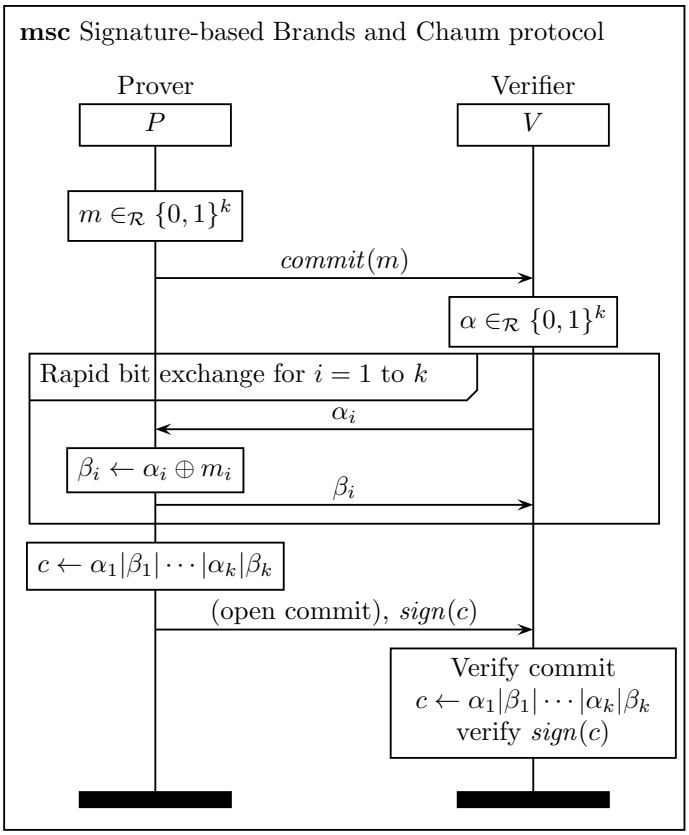

Figure 1: Signature-based Brands and Chaum protocol.

Desmedt [8]. In this type of attack, both the prover P and verifier V are honest, and the attack is performed by an external attacker $\mathcal{A}$. The attacker attempts to shorten the distance measured between the honest prover and the verifier. In Mafia Fraud attacks, the physical distance between the attacker and the verifier is typically small in order for the attacker to be able to shorten the distance.

In a *Distance Fraud* attack, a dishonest prover P will try to shorten the distance measured by the verifier V. This type of attack is executed by the dishonest prover P alone, without collusion with other (external) parties. An example of a Distance Fraud attack occurs if the protocol allows the prover to send his reply before receiving the challenge. This enables the prover to reply too early, thereby shortening the distance measured by the verifier.

The third class of attacks is *Terrorist Fraud* attacks [8]. In this type of attack, a dishonest prover P collaborates with an external attacker $\mathcal{A}$ to convince the verifier V that he is closer than he really is. All countermeasures to Terrorist Fraud make the assumption that the dishonest prover P is unwilling to reveal

his long-term (private or secret) key to the attacker $\mathcal{A}$ that he collaborates with. Possible reasons for this unwillingness are impersonation, i. e., the external attacker can later use the key to impersonate the dishonest prover, and traceability, i. e., the key may later be used to implicate the dishonest prover in performing a Terrorist Fraud attack. Furthermore, from the perspective of the verifier, it is impossible to distinguish between the external attacker and the prover if the attacker knows the long term key of the prover.

3 Distance Hijacking

In this section we define a fourth class that has until now been overlooked in the design of distance bounding protocols, *Distance Hijacking attacks*. We relate this class of attacks to the three classical attack types on distance bounding protocols, and propose an exhaustive classification of attacks on distance bounding protocols.

3.1 Distance Hijacking attacks

We say that a prover P is *honest* if and only if all of P's actions conform to the protocol specification.

Definition 1. *A* Distance Hijacking attack *is an attack in which a dishonest prover P exploits one or more honest parties $P_1, \ldots, P_n$ to provide a verifier V with false information about the distance between P and V.*

A protocol is then said to be vulnerable to Distance Hijacking if it allows P to perform a successful Distance Hijacking attack. We observe that these attacks do not exclude the involvement of other attackers with whom the dishonest prover is colluding or the involvement of other honest verifiers that might enable the execution of the attack.

In the context of distance bounding protocols, the information about the distance is the upper bound; hence attacks involve convincing V that P is closer than it actually is. In a typical Distance Hijacking attack on a distance bounding protocol, a dishonest prover P convinces a verifier V that P has executed a distance measurement phase (e. g., a rapid bit exchange) with V, whereas this phase has been really executed by an honest prover P'. This is done without the cooperation of the honest prover P'. Often this type of attack can be carried out by allowing the honest prover to complete the distance bounding protocol as he normally would, and then by replacing all messages that contain signatures or MACs, with messages signed (or MAC'ed) by the attacker.

Example 1 (Distance hijacking attack on signature-based Brands and Chaum)**.** *Figure 2 depicts a basic Distance Hijacking attack on the signature-based Brands and Chaum from Figure 1.*

In the attack, V thinks he is communicating with P, where P is dishonest. When an honest prover P' tries to prove his distance, P initially allows the

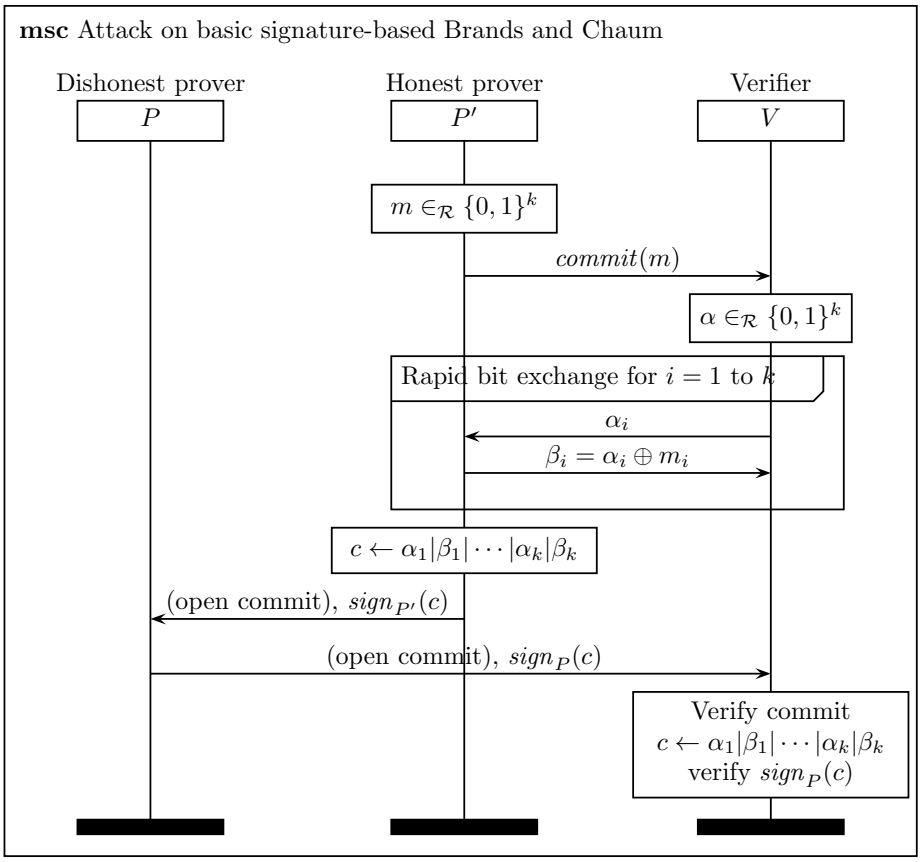

Figure 2: Distance Hijacking attack on basic signature-based Brands and Chaum. $sign_P$ and $sign_{P'}$ denote the signatures with the signature keys of P and P', respectively.

protocol to proceed as normal between P' and V, waiting until the final signature is sent by P'. Note that before this point, V has no cryptographic evidence that the messages it received were indeed sent by P'. When P' sends the signature, P jams the message and re-signs the content c with his own signature key, and sends the result to V. V will successfully verify the commit as well as the signature, and will falsely conclude that P has also sent the previous message. Thus, V assumes that P is within the distance computed from the distance bounding phase, even though in reality, this phase was performed by P'.

We next show an example scenario in which Distance Hijacking attacks pose a threat.

Example 2 (Real-world scenario)**.** *Consider the scenario depicted in Figure 3, in which several people work in a secure facility. In the facility is a mainframe*

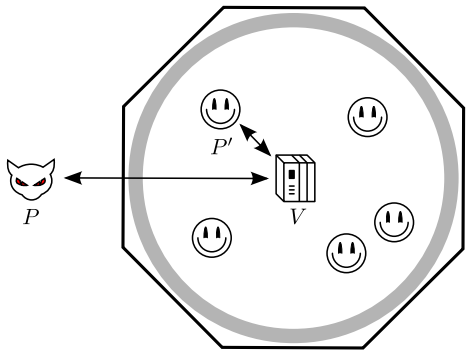

Figure 3: Real-world scenario for Distance Hijacking. P has a (stolen) smartcard. However, he cannot enter the secure facility and he does not have any collaborators inside the facility. In a Distance Hijacking attack, P exploits the presence of an honest P' to convince V that P is within the secure facility.

containing sensitive information. The mainframe can be accessed wirelessly by all authorized personnel, in order to facilitate easy access by multiple people at the same time. As an added security mechanism, in case an employee loses his smartcard with his private key, the mainframe can only be accessed by people inside the building. This is verified every time an employee logs in to the system, by running a distance bounding protocol between a station in the building (acting as the verifier) and the employees terminal (acting as the prover).

Assume that an attacker has managed to get hold of an employee smartcard but is unable to physically access the building. He is instead located in a van in the parking lot where he has a powerful antenna capable of communicating with the wireless terminal inside the building. However in order to log in to the system the attacker needs to prove that he is inside the building by running a distance bounding protocol.

If the distance bounding protocol in use is vulnerable to distance hijacking, the attacker can exploit the presence of the smartcard of another (non-collaborating and unaware) employee inside the building to execute a Distance Hijacking attack. The mainframe security system now believes that the attacker is in the building with a valid private key, and he is granted wireless access.

As straightforward as this type of attack may seem, a surprising number of distance bounding protocols are vulnerable to Distance Hijacking, as we will show in Section 3.5. In Section 6 we discuss more complex Distance Hijacking attacks, where several different distance bounding protocols are used in the same environment.

3.2 Relation to historical attack types

We first relate Distance Hijacking to the three attack types that are traditionally considered for distance bounding protocols.

As stated in the introduction, conceptually speaking, Distance Hijacking can be placed between Distance Fraud and Terrorist Fraud. One could thus consider extending the definition of either Distance Fraud or Terrorist Fraud to also include Distance Hijacking attacks. However, given that previous analyses and countermeasures do not exclude such attacks, the consequence would be that many protocols would be incorrectly labeled as being resistant against the (new definitions of) Distance Fraud or Terrorist Fraud, or that existing countermeasures are insufficient. We therefore choose to introduce Distance Hijacking as a separate type of attack.

We show why the existing three attack types do not cover Distance Hijacking. In Mafia Fraud attacks the prover is honest. Distance Fraud attacks are defined as attacks by a lone dishonest prover. These two types are therefore clearly different from Distance Hijacking, which involves at least a dishonest prover and another honest party.

To illustrate the difference between Distance Hijacking and the attack type that is conceptually closest, Terrorist Fraud, we consider again the scenario from Example 2. Recall that in Terrorist Fraud, the dishonest prover collaborates with another (closer) attacker. In the scenario from the example, there are two main reasons why the absence of Terrorist Fraud attacks does not guarantee the absence of Distance Hijacking attacks. First, we observe that Terrorist Fraud is not possible in this scenario, because the attacker does not have another attacker inside the building that is willing to cooperate with him. Hence the designers of the system could consider using a protocol such as signature-based Brands and Chaum, on which Distance Hijacking may still be possible. Second, the common countermeasure to Terrorist Fraud is to force the attacker to reveal his long term key to his accomplice, based on the assumption that this will deter the attacker. However, in the scenario from Example 2 this assumption does not hold: the attacker has no problem with leaking the (stolen) long term key. Additionally, even if he does transmit the key, it will be to the (unmodified) smartcard of an honest employee. The employee's smartcard will typically not detect this key, and will even delete the received data after the session ends. Hence guaranteeing the absence of Terrorist Fraud attacks, either by assumption or by countermeasure, does not guarantee the absence of Distance Hijacking.

3.3 Attack classification

The traditional attack types Mafia Fraud, Distance Fraud, and Terrorist Fraud, are defined independent of each other and usually in incompatible ways. This makes it hard to determine whether all possible attacks on distance bounding protocols are covered by these types, even if we include Distance Hijacking attacks. We propose to remedy this situation by deriving attack type definitions that cover all possible attacks by construction. Intuitively, we perform a sequence of case distinctions based on three attributes of attacks on distance bounding protocols: whether the prover is honest, whether the prover is the only party involved in attacking the verifier, and if not, whether one of the other involved parties is honest. By considering these three attributes we arrive at definitions

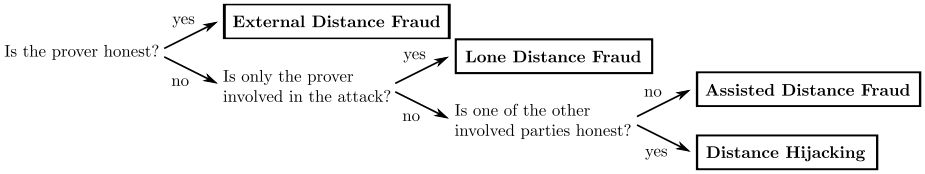

Figure 4: Classification of attacks on distance bounding protocols, in which a verifier computes an incorrect distance bound for a prover.

for four attack types.

We introduce some additional terminology. The goal of a distance bounding protocol is to compute a correct distance bound. More precisely, we say that the verifier V computes the *correct distance bound* d on P, if P or his identifying key[1] is indeed within the (computed or expected) distance d. We make two assumptions on distance-bounding protocols. First, in the absence of attackers, the verifier computes the correct distance bound. Second, we assume that the protocols guarantee weak authentication of P (i. e., *aliveness* [17]).

Using the above terminology and assumptions, we provide an exhaustive classification of attacks on distance bounding protocols attacks in which the verifier computes an incorrect distance bound for the prover, represented in Figure 4. Assume that V does not compute the correct distance bound d for P. Thus, neither P nor his identifying key is within the distance d. Because of our first protocol assumption, this must be caused by an attacker.

We distinguish two main cases. If P is honest, then P is not the attacker, and therefore an external attacker is changing the distance. We call this type of attack *External Distance Fraud*.

Definition 2. *An* External Distance Fraud attack *is an attack in which an attacker provides a verifier V with false information about the distance between an honest prover P and V.*

In the second case, if P is not honest, then we distinguish again between two cases. First, if only P is involved in the attack, he must be the attacker, trying to change his own distance. We call this type of attack *Lone Distance Fraud*.

Definition 3. *A* Lone Distance Fraud attack *is an attack in which a lone prover P provides a verifier V with false information about the distance between P and V.*

If other parties are involved, we make a final distinction. If all of the other parties are dishonest or collaborating, the attack is called an *Assisted Distance Fraud* attack.

Definition 4. *An* Assisted Distance Fraud attack *is an attack in which a prover P is assisted by one or more other parties, none of which are honest, to provide a verifier V with false information about the distance between P and V.*

[1] In our context, P is identified by his key. If others know P's key, they cannot be distinguished from P.

Alternatively, if one of the other parties involved in the attack is honest, we call the attack a *Distance Hijacking* attack, as in Definition 1.

In constructing the above classification, we have tried to stay close to the historical attack types. In fact, three of our attack types are variants of the historical types. However, we have tried to provide them with more descriptive and less generic names. In particular, our definition of Lone Distance Fraud closely resembles the classical notion of Distance Fraud. Our definition of External Distance Fraud resembles that of Mafia Fraud, and our definition of Assisted Distance Fraud includes Terrorist Fraud attacks.

It is worth pointing out that although P refers to a specific identity, or rather the identity of a party holding a specific key, this classification is also valid in the context of anonymous distance bounding protocols [30]. In anonymous distance bounding, the only guarantee provided to the verifier is that *someone* is within a specific distance, as opposed to P is within a specific distance. In order to fit anonymous distance bounding protocols into this model, we say that all provers in anonymous distance bounding share the same key (which could be public) and, in the decision points in Figure 4, "the prover" must be replaced by "the closest prover".

3.4 Multi-prover environments

The main requirement for Distance Hijacking is that there are other parties in the environment, which can be exploited by a dishonest prover. We call environments in which multiple provers may occur *multi-prover environments*. We give two concrete examples of such environments.

Multiple provers, single verifier One such a scenario occurs when a verifier accepts proofs from multiple provers, as depicted in Figure 5. For example, this may occur in RFID distance bounding where a reader may accept multiple tags. In this case, Distance Hijacking occurs when a dishonest prover P hijacks the distance from P' to V and instead convinces V that P is at this distance, thereby falsely "shortening" the distance between P and V.

Note that in the above example, the verifiers accept protocol sessions from multiple provers. Below we show that this is not required for the attacks.

Multiple provers, multiple verifiers Consider an environment with multiple provers $P, P', \ldots$ and corresponding verifiers, $V_P, V_{P'}, \ldots$, where verifier V_P only accepts proofs-of-distance from prover P and verifier $V_{P'}$ only from prover P'. Even in this scenario, a prover P can hijack a session from a prover P' to a verifier $V_{P'}$ to make V_P falsely believe that P is at distance $dist(P', V_P)$. This type of scenario is depicted in Figure 6. P' assumes that he is proving his distance to $V_{P'}$, but instead, the fast response of P' is accepted by V_P, who assumes that it was sent by P.

Note that for the attack to work, neither P and V_P nor P and P' need to be physically close. Instead, the communication between P' and $V_{P'}$ can

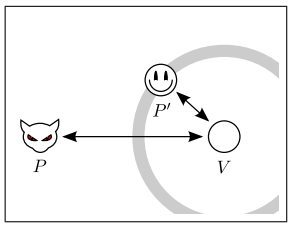

Figure 5: Scenario in which V accepts protocol sessions from multiple provers, here P and P', where Distance Hijacking may be a threat.

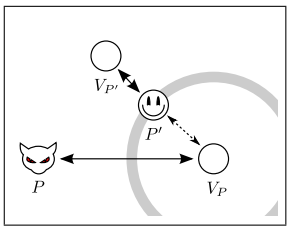

Figure 6: Scenario with multiple prover/verifier pairs, where V_x only accepts sessions from x. Even in this case, Distance Hijacking may be possible.

be enabled by the attacker who created a relay between them whereas P can communicate to P' using a high power transceiver and a high gain antenna. This second scenario may even occur across domains: the only requirement is that the distance measurement (e. g., rapid bit exchange) phases used in both domains are to some extent compatible.

3.5 Analysis of Existing Distance Bounding Protocols

We have analyzed several protocols and found numerous new attacks that fall into the class of Distance Hijacking attacks. We give an overview of the protocols analyzed in Table 1. The vast majority of the attacks we find are new. To the best of our knowledge, only two such attacks were previously reported in the literature. The attack on a simplified version of "Brands and Chaum (signature)" is described in [25]. The attack on a member of the protocol family proposed by Meadows et al., in particular for the instance with $F(NV, NP, P) = \langle NV, NP \oplus P \rangle$, is described in [3]. All other attacks in the table are new.

In our analysis we used the following system and attacker model. We assume that the attacker controls the network and may eavesdrop, intercept, inject, and block messages. We do not pose any restrictions on the number or locations of devices that the attacker holds; the attacker can control several dishonest provers as well as other wireless devices. Entities are identified by their keys; entities that hold the same keys cannot be distinguished.

In this paper, we describe two attacks from the table in detail. We already described the attack on the basic Brands and Chaum protocol with signatures in Example 1. We describe an attack on the Kuhn, Luecken, Tippenhauer protocol in Example 4.

In general, it seems that protocols that closely follow the original Brands and Chaum protocols do not offer protection against Distance Hijacking. In contrast, protocols that derive from the Hancke and Kuhn protocol, which explicitly uses the key shared between agents in the distance bounding phase, protect against Distance Hijacking in single-protocol environments. However, as we explain in Section 6, all protocols, including the ones derived from the Hancke and

No.	Protocol	Discovered attack
1	Brands and Chaum (Fiat-Shamir) [4, p. 351]	Yes
2	Brands and Chaum (Schnorr) [4, p. 353]	Yes
3	Brands and Chaum (signature) [4, p. 350]	Yes
4	Bussard and Bagga [5]	No
5	CRCS [23]	Yes
6	Hancke and Kuhn [12]	No
7	Hitomi [20]	No
8	KA2 [15]	No
9	Kuhn, Luecken, Tippenhauer [16]	Yes
10	MAD [29]	Yes
11	Meadows et al. for $F(\ldots) = \langle NV, NP \oplus P \rangle$ [18]	Yes
12	Munilla and Peinado [19]	No
13	Noise resilient MAD [27]	Yes
14	Poulidor [28]	No
15	Reid et al. [24]	No
16	Swiss-Knife [14]	No
17	Tree [2]	No
18	WSBC+DB [21, p. 50]	Yes
19	WSBC+DB Noent [21, p. 51]	Yes

Table 1: Discovered Distance Hijacking attacks on existing protocols (single protocol environment).

Kuhn protocol, are vulnerable to Distance Hijacking in specific multi-protocol environments.

We note that for many of the attacks in the table, it is required that the verifier V is not "disturbed" by P's messages. As a concrete example, consider the attack in Figure 2. If V would receive and parse P's final signed message, V might abort the protocol, in which case the attack fails. There are several practical scenarios in which the attacks are directly possible. For example, assume that the signed message is sent through standard WiFi channels, and P assumes that he is responding to some other verifier V_2. In this case, P sends the message addressed to V_2, and V's hardware may already filter out the message before it arrives at the protocol level. Alternatively, the attacker can jam-and-eavesdrop the signals sent by P (except for P's fast response). Jamming seems to be possible on all protocols in the table except for MAD, which explicitly requires jamming detection, upon which the protocol aborts.

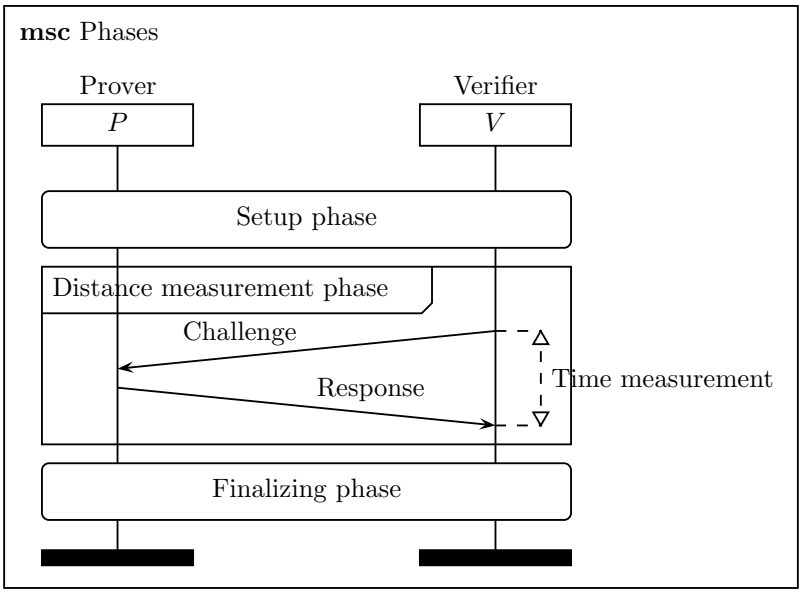

Figure 7: Phases in distance bounding protocols: Setup, distance measurement, and finalizing. The setup and finalizing phases may be empty.

4 Protecting against Distance Hijacking

We have seen that many protocols are vulnerable to Distance Hijacking, and we now show how to repair them. Without loss of generality, any distance bounding protocol can be divided into three phases as depicted in Figure 7: the *setup phase*, where nonces and commitments are exchanged; the *distance measurement phase*, where the physical distance is measured, often using rapid bit exchange; and the *finalizing phase* that often includes a proof of identity. The only phase that is required to be non-empty is the distance measurement phase. The distance measurement phase follows the following schema: the verifier sends out a fresh challenge, to which the prover responds with some value; this process may be split into several rounds. The distance measurement is derived from the measured response time, which means that the prover must reply immediately. It is therefore infeasible to use cryptographic functions (such as encryption or signatures) in the computation of the response in this phase.

In a typical Distance Hijacking attack, a dishonest prover exploits another prover's response in the distance measurement phase. Thus, although the dishonest prover has few restrictions, because he does not have to follow the protocol and can construct his own messages as he chooses, he can only exploit honest provers as far as the protocol allows him to. Therefore, in the fixes we propose, we ensure that the distance measurement response of an honest prover cannot be abused by others in their communication with the verifier.

Before we proceed to solutions, we provide more intuition by showing why

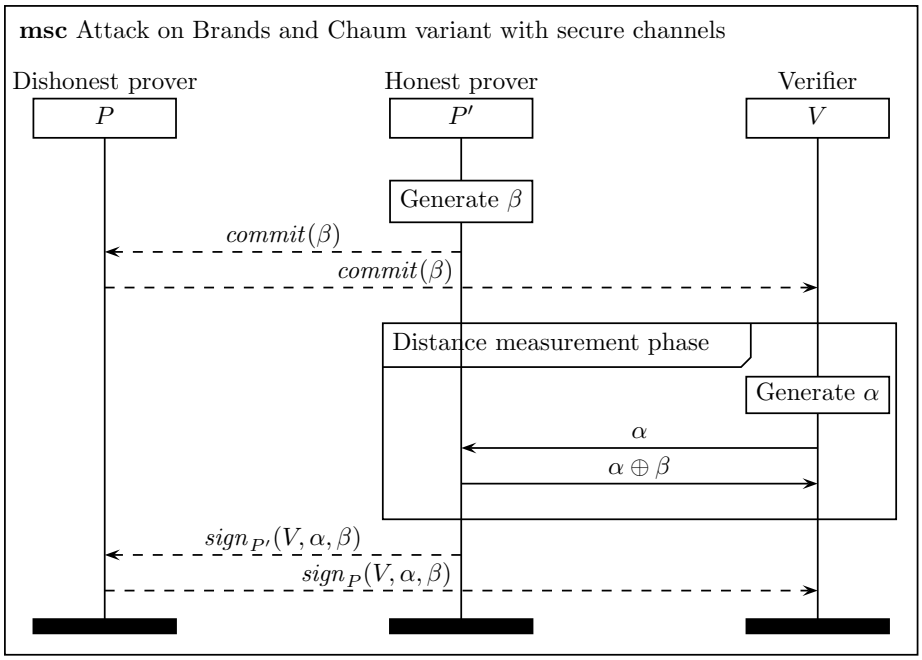

Figure 8: Attack on Brands and Chaum variant where setup and finalization use a secure channel. We use dashed arrows to denote transmission over a secure channel. P' assumes that P is a verifier.

two seemingly straightforward fixes to the basic Brands and Chaum protocol fail.

Example 3 (Flawed fix: Xor identity)**.** *A first flawed fix is to include the prover's identity in the response messages by sending challenge* $\oplus$ *NP* $\oplus$ *P. The problem with this solution is that the identity of an attacker* P' *might only differ in a few bits from* P*. Then challenge* $\oplus$ *NP* $\oplus$ *P agrees with challenge* $\oplus$ *NP* $\oplus$ P' *on all other bits and the adversary only has to guess the remaining few bits in challenge* $\oplus$ *NP* $\oplus$ P' *and overshadow them in P's response. After learning challenge and NP, the dishonest prover can check if his guesses were right and send the final signature. If the Hamming distance between* P *and* P' *is* k*, then the attacker has to guess* k *bits and his success probability is therefore* $(\frac{1}{2})^k$*.*

Example 4 (Flawed fix: secure channels)**.** *A second fix is to perform the setup and finalizing phases over some secure channel, e. g., by using SSL/TLS, mutually authenticated using client and server certificates. A protocol along these lines is described in [16]. Because an attacker now cannot eavesdrop (or change) the contents of the communication, it might seem that any hijacking is thwarted. However, as depicted in Figure 8, such protocols are still vulnerable to Distance Hijacking. In the attack,* P *claims to be a verifier when communicating with* P'*, and* P *claims to be a prover when communicating with* V*. Thus,* P' *assumes*

that he is proving his distance to P, and therefore transmits his commit over the secure channel to P. P simply forwards this commit to V. Because the distance measurement phase is not protected by the secure channel, P' will respond to V's challenge. Afterwards, P' will finalize his part over the secure channel with P. P re-signs this information and sends it to V over the secure channel.

As shown by the examples, it is not trivial to make protocols resilient against Distance Hijacking. The solution is to make the prover's messages during the distance measurement phase distinguishable from those of other provers, such that a verifier will not mistake the response of one prover (say, P') for the response of another (say, P). We discuss two possible solutions: explicit linking and implicit linking.

Solution family 1: Explicit linking The first solution, *explicit linking*, ensures that the response from different provers is distinguishable by explicitly including identity information in the response, combined with integrity protection. Example instances of explicit linking are the following, where we assume that NP is a nonce generated by the prover which he commits to in the setup phase.

- $challenge \oplus h(P, NP)$, where h is a hash function.
- $challenge \oplus sign_P(NP)$.
- $challenge \oplus MAC_{k(P,V)}(P, NP)$, where $k(P, V)$ is a symmetric shared key between P and V.

Solution family 2: Implicit linking The second solution type, *implicit* linking, does not make the responses of different provers distinguishable on their own. Rather, it relies on the fact that honest provers do not reveal some secret, typically their own nonce NP, before the distance measurement phase has been completed. Thus, before this phase, only the prover who generated NP knows the secret and can use it to construct messages. In protocols that commit to a (temporary) secret in the setup phase, the prover can include his identity in the commit, hence sending $commit(P, NP)$ before the distance measurement phase. Until the prover P releases this nonce during or after his response, other (dishonest) provers cannot commit to NP with their own identity. Thus, the verifier can check that the claimed identity for the distance measurement phase corresponds to the commit he received during the setup phase.

5 Formal Analysis

Previous formal models capture Distance Hijacking to an insufficient extent. Specifically, they do not capture *overshadowing* parts of a message (see [22]), e. g., by sending bits using a stronger signal. Several of our Distance Hijacking attacks involve such overshadowing. To capture these attacks, we extend the formal framework of Basin et al. [3] to allow the attacker to perform message

manipulation on the wireless channel by overshadowing parts of a message, as well as flipping some bits of a message. The resulting new framework allows to formally prove the absence all of the previously described Distance Hijacking attacks. A complete Isabelle/HOL formalization of all definitions and proofs in this section is available in [26].

In Section 5.1, we recall the basic model from [3] and we present its extension in Section 5.2.

5.1 Basic model

Agents and Environment We assume that there are countably infinite disjoint sets *Honest* and *Dishonest* of honest and dishonest agents. We define the set of all agents as $Agent = Honest \cup Dishonest$. We use A, B, P, V for agents. We associate a location $loc_A \in \mathbb{R}^3$ to each agent. Based on the location, we define the line-of-sight communication distance between two agents A and B as

$$cdist_{LoS}(A, B) = \frac{|loc_A - loc_B|}{c}$$

where c denotes the speed of light. This distance constitutes a lower bound on the time required for a signal to travel from A to B derived from the locations of both agents.

Messages We assume that there is a countably infinite set *Const* of constants. We assume that there are countably infinite disjoint sets $Nonce_A$ for each agent A and define $Nonce = \bigcup_{A \in Agent} Nonce_A$. We assume that there is a countably infinite set *Key* of keys that is partitioned into keys for symmetric encryption and asymmetric encryption/signatures. We assume that there is an inverse operator $\cdot^{-1}$ on *Key* that is the identity on symmetric keys. The set of *syntactic messages* *SMsg* is defined by the grammar

$$M, M' ::= atom \mid \langle M, M' \rangle \mid h(M) \mid \{M\}_k \mid M \oplus M' \mid 0$$

where $atom \in Agent \cup Const \cup Key \cup Nonce$ is an atomic message, and the remaining cases denote pairing, hashing, encryption with $k \in Key$, exclusive-or, and the all-zero message. We write $sign_A(M)$ as a shorthand for $\{M\}_{sk(A)}$.

We define the set *Msg* of *messages* as $SMsg/=_E$, where $=_E$ is the equational theory generated by the set of equations

$$\begin{aligned} E = \{&M \oplus 0 = M, M \oplus M = 0, \\ &(M \oplus M') \oplus M'' = M \oplus (M' \oplus M''), \\ &M \oplus M' = M' \oplus M\}. \end{aligned}$$

In the following, we abuse notation and write M to denote the corresponding equivalence class $\{M' \mid M' =_E M\} \in Msg$.

$$\frac{M \in IK_A}{M \in DM_A(tr)} \qquad \frac{(t, Recv_A(M)) \in tr}{M \in DM_A(tr)} \qquad \frac{M \in DM_A(tr) \quad M' \in DM_A(tr)}{M \oplus M' \in DM_A(tr)}$$

$$\frac{M \in DM_A(tr) \quad M' \in DM_A(tr)}{\langle M, M' \rangle \in DM_A(tr)} \qquad \frac{M \in DM_A(tr)}{h(M) \in DM_A(tr)}$$

$$\frac{M \in DM_A(tr) \quad k \in DM_A(tr)}{\{M\}_k \in DM_A(tr)} \qquad \frac{\langle M_1, M_2 \rangle \in DM_A(tr)}{M_i \in DM_A(tr)}$$

$$\frac{\{M\}_k \in DM_A(tr) \quad k^{-1} \in DM_A(tr)}{M \in DM_A(tr)}$$

Figure 9: Rules defining $DM_A(tr)$.

Events and Traces The set of events is defined as

$$EV ::= Send_A(M)[M^*] \mid Recv_A(M) \mid Claim_A(M).$$

For *Send*, A denotes the agent executing the send, M the sent message, and M^* is a sequence of messages denoting local state information associated with the event. For *Recv*, A denotes the agent executing the receive and M the received message. For *Claim*, A denotes the agent making the claim and M the claim itself. A trace tr is a sequence of timed events (t, EV) with $t \in \mathbb{R}$.

Initial knowledge To model initial key distributions, we define the functions $pk : Agent \to Key$, $sk : Agent \to Key$, and $K : Agent \times Agent \to Key$ that denote the public, secret, and shared keys of agents with the expected properties, e. g., $pk(A)^{-1} = sk(A)$ and $K(A, B) = K(B, A)$. We define the initial knowledge of an agent A as

$$\begin{aligned} IK_A = {} & Agent \cup Const \cup Nonce_A \cup \{0\} \\ & \cup \{sk(A)\} \cup \{pk(B) \mid B \in Agent\} \\ & \cup \{K(A, B) \mid B \in Agent\}. \end{aligned}$$

Message deduction Let A be an agent and let tr be a trace. Then the set $DM_A(tr)$ of *deducible messages* is the least set closed under the rules in Figure 9. The rules model message manipulations under the perfect cryptography assumption, and are all considered modulo E.

Network and Attacker The set of possible traces TR for the basic model is defined as the least set closed under the Start-rule, the attacker rule Intr, and the basic network rule BasicNet[2] given in Figure 10 and the rules formalizing

[2]Note that this rule is called Net in [3].

$$\frac{}{\epsilon \in TR}\ \textsc{Nil} \qquad \frac{tr \in TR \quad I \in Dishonest \quad M \in DM_I(tr)}{tr \cdot (t, Send_I(M)[]) \in TR_P}\ \textsc{Intr}$$

$$\frac{tr \in TR \quad (t', Send_A(M)[L]) \in tr \qquad t \geq t' + cdist_{LoS}(A, B)}{tr \cdot (t, Recv_B(M)) \in TR_P}\ \textsc{BasicNet}$$

Figure 10: Rules for network and attacker from the basic model.

$$\frac{\begin{array}{l} tr \in TR \end{array} \quad \begin{array}{l} \forall\, X \in components(M). \\ \quad \exists\, t'\ A\ L\ M'\ Y \in components(M'). \\ \qquad (t', Send_A(M')[L]) \in tr \\ \qquad \wedge\, X \oplus Y \in LHW \\ \qquad \wedge\, t \geq t' + cdist_{LoS}(A, B) \end{array}}{tr \cdot (t, Recv_B(M)) \in TR_P}\ \textsc{ExtNet}$$

Figure 11: The new network rule for the extended model.

the analyzed protocol. For an example of protocol rules, see Figure 12. All rules have the implicit side condition that timestamps are monotonous, i. e., the timestamp of a newly added event cannot be smaller than the maximal timestamp in the trace. The Intr rule allows dishonest agents to send arbitrary deducible messages. The BasicNet rule formalizes that if there is a message M that has been sent by an agent A, then B can receive the message at time t if $t \geq t' + cdist_{LoS}(A, B)$.

Given a set of traces of a protocol for a model, we can define when a protocol is secure.

Definition 5. *A distance bounding protocol is secure if all claims $(V, P, dist)$ that occur in traces of the protocol are valid, i. e., they agree with loc_V and loc_P. Here, we account for the fact that we allow dishonest nodes to share key material and therefore identify all dishonest agents, i. e., a claim $(V, P, dist)$ for dishonest P is valid if there is some dishonest P' such that $dist$ is an upper bound on the distance between V and P'.*

5.2 Extended model

The network rule BasicNet from [3] does not account for message manipulation on the wireless channel. As a result, several attacks from the previous sections (e. g., the attack in Example 3) cannot be reproduced in the basic model. We define our extended model by replacing the network rule BasicNet by a new

rule ExtNet, shown in Figure 11, that allows for a more fine-grained model of message manipulation.

We start from the observation that the BasicNet rule does not account for the ability of an attacker to overshadow parts of a message. Overshadowing can be used to replace components in pairs with known messages or to transform an (unknown) message M into an (unknown) message M' if M and M' differ only in a few bits.

In the context of a model where signals traverse distances, as required for modeling distance bounding protocols, an attacker that is far from a receiver (e. g., the verifier) may want to use overshadowing to modify the message M of a sender (e. g., the prover) that is close to the intended recipient into a message M'. Let t be the time at which the sender sends the message M. Because the attacker is further away, the attacker needs to send the overshadowing bits $(M \oplus M')$ at time t', where $t' < t$, to ensure that they arrive at the same time at the receiver. If the attacker knows the (parts of the) message that he wants the recipient to receive, this is straightforward. However, if the attacker does not know the message M yet at time t' and requires the message M to compute M', he needs to guess the positions where M and M' differ, then guess the bits of M' on these positions, and finally overshadow M on these positions with the guessed bits. We assume that guessing many of the bits of M correctly can only be done with negligible probability. Subsequently, the attacker can transform M into M' with non-negligible probability if and only if the Hamming distance between M and M' is small.

To account for these manipulations, we require two definitions. First, the *components* of a message M are defined as $\mathit{components}(M) = \mathit{components}(M_1) \cup \mathit{components}(M_2)$ if $M = \langle M_1, M_2 \rangle$ and $\mathit{components}(M) = \{M\}$ otherwise. Second, the set LHW of messages that may have a low Hamming weight is defined as

$$L, L' ::= \mathit{latom} \mid L \oplus L' \mid 0$$

where $\mathit{latom} = \mathit{Agent} \cup \mathit{Const}$. This excludes nonces, keys, hashes, encryptions, and the exclusive-or of such messages since the probability that these have a low Hamming weight can be assumed to be negligible, unless such a message cancels itself out. We do not include pairs of low Hamming weight messages since we already allow the attacker to modify components of pairs individually.

Our new network rule ExtNet is shown in Figure 11. According to the rule, an agent B can receive a message M if for all components X of M, there is a corresponding send event (with compatible timestamp) of a message M' such that M' has a component Y with a low Hamming distance to X, i. e., the Hamming weight of $X \oplus Y$ is low.

Example 5. *We assume that the attacker does not know NV and NP. To overshadow NP with NI in the message $\langle NV, NP \rangle$ sent by an honest P, the attacker has to send NI (early enough) such that both sends together result in a receive of $\langle NV, NI \rangle$.*

In Example 3, the attacker overshadows some bits to transform the (unknown) message $NV \oplus NP \oplus P$ into the (unknown) message $NV \oplus NP \oplus P'$. In our

$$\frac{tr \in TR \quad P \in Honest \qquad NP \in (Nonce_P \setminus subterms(tr))}{tr \cdot (t, Send_P(h(NP))[\mathcal{P}_1, NP]) \in TR}\ \textsc{ProvCom}$$

$$\frac{\begin{array}{c} tr \in TR \quad V \in Honest \\ (t, Recv_V(COM)) \in tr \qquad NV \in (Nonce_V \setminus subterms(tr)) \end{array}}{tr \cdot (t, Send_V(NV)[\mathcal{V}_1, COM, NV]) \in TR}\ \textsc{VerChal}$$

$$\frac{\begin{array}{c} tr \in TR \quad P \in Honest \\ (t, Recv_P(NV)) \in tr \qquad (t, Send_P(X)[\mathcal{P}_1, NP]) \in tr \end{array}}{tr \cdot (t, Send_P(NV \oplus NP)[\mathcal{P}_2, NP, NV]) \in TR}\ \textsc{ProvResp}$$

$$\frac{tr \in TR \quad P \in Honest \qquad (t, Send_P(X)[\mathcal{P}_2, NP, NV]) \in tr}{tr \cdot (t, Send_P(sign_P(NV, NP, P))[]) \in TR}\ \textsc{ProvAuth}$$

$$\frac{\begin{array}{c} tr \in TR \quad V \in Honest \qquad (t_{chal}, Send_V(NV)[\mathcal{V}_1, h(NP), NV]) \in tr \\ (t_{resp}, Recv_V(NV \oplus NP)) \in tr \\ (t_{auth}, Recv_V(sign_P(NV, NP, P))) \in tr \end{array}}{tr \cdot (t, Claim_V(V, P, (t_{resp} - t_{chal}) * c/2)) \in TR}\ \textsc{VerResp}$$

Figure 12: Formalization of the Brands-Chaum Protocol.

model, the attacker does not have to perform any action since $(NV \oplus NP \oplus P') \oplus (NV \oplus NP \oplus P) = P \oplus P' \in LHW$. *This captures the intuition that allowing the attacker to flip some bits of unknown messages is equivalent to allowing for some bit-errors introduced by the wireless channel.*

The set of possible traces TR for our extended model is defined as the least set closed under the Start-rule, the attacker rule Intr, the extended network rule ExtNet, and the rules formalizing the analyzed protocol.

Protocol Formalization We formalize the original signature-based version of the Brands-Chaum by the rules in Figure 12. $\mathcal{P}_i$ and $\mathcal{V}_i$ are constants used in the local state of the verifier and prover in step i. The rules ensure that the previous steps have been executed, the required messages have been received, and nonces are freshly chosen (not subterm of the trace tr). The final rule VerResp uses the times when the challenge was sent and the time when the reply was received to compute an upper bound on the distance between P and V. We refer the reader to [3] to further details on modeling protocols in this kind of framework.

Case studies We have analyzed the Brands-Chaum protocol and its various fixes in our extended framework. For example, we have proven that if we modify Brands-Chaum to include explicit linking, the Distance Hijacking attack is no longer possible. Note that for proving the correctness of the version with implicit

linking, we need the assumption that verifiers cannot receive the bits/message that they sent themselves, e. g., because different channels are used. Without this assumption, Brands-Chaum is vulnerable to Distance Fraud attacks (by reflection or using low Hamming weight overshadowing). As another example, our extended framework also reveals that modifying the response to $NV \oplus NP \oplus P$ is not secure since the attack from Example 3 is captured. Note that in the basic framework from [3], which did not account for message manipulation on the wireless channel, this attack was not captured and a security proof was possible. This clearly shows the effect of the adversary's additional powers in our extended model.

For a complete description of our case studies and their formalization we refer the reader to [26].

6 Multi-protocol Environments

So far, we discussed Distance Hijacking attacks in *single-protocol environments*, where both dishonest and honest prover run the same distance bounding protocol. However, it is possible that verifier-prover pairs execute different ranging and distance bounding protocols, for example when they belong to different domains. We call such environments *multi-protocol environments*.

Distance Hijacking in Multi-protocol Environments In what follows we show that there are plausible multi-protocol environments in which protocols that are resilient to Distance Hijacking in single-protocol environments become vulnerable again to Distance Hijacking.

We define a multi-protocol environment MPE as a set of triplets, where a triplet (A, B, R) denotes that agent A may execute the protocol role R (e. g., the prover role of the Brands and Chaum protocol) when communicating with B, and where at least two different protocols are contained in the set. We say that a distance bounding protocol DB is vulnerable to a Distance Hijacking Attack in a multi-protocol environment MPE if a dishonest prover P can perform a successful Distance Hijacking attack against a verifier V running DB in the verifier role in that environment (and hence $(V, P, DB(\text{verifier})) \in MPE$).

It is easy to see that, given *any* distance bounding protocol, a multi-protocol environment can be constructed in which this protocol will be vulnerable to Distance Hijacking attacks. For example, all distance bounding protocols will be vulnerable to Distance Hijacking if run in the same environment with a protocol that uses a similar distance measurement phase, but that gives a dishonest prover full control over the way the response bits are computed by the honest prover. This is not such an unlikely scenario, since it is plausible that in the same environment in which a verifier and a dishonest prover run e. g., Hancke and Kuhn, an honest prover runs an insecure ranging protocol that supports the same type of distance measurement phase as the Hancke and Kuhn protocol. This insecure ranging protocol could easily allow a dishonest prover to set the bits that the honest prover uses in the distance measurement phase (e. g., for debugging

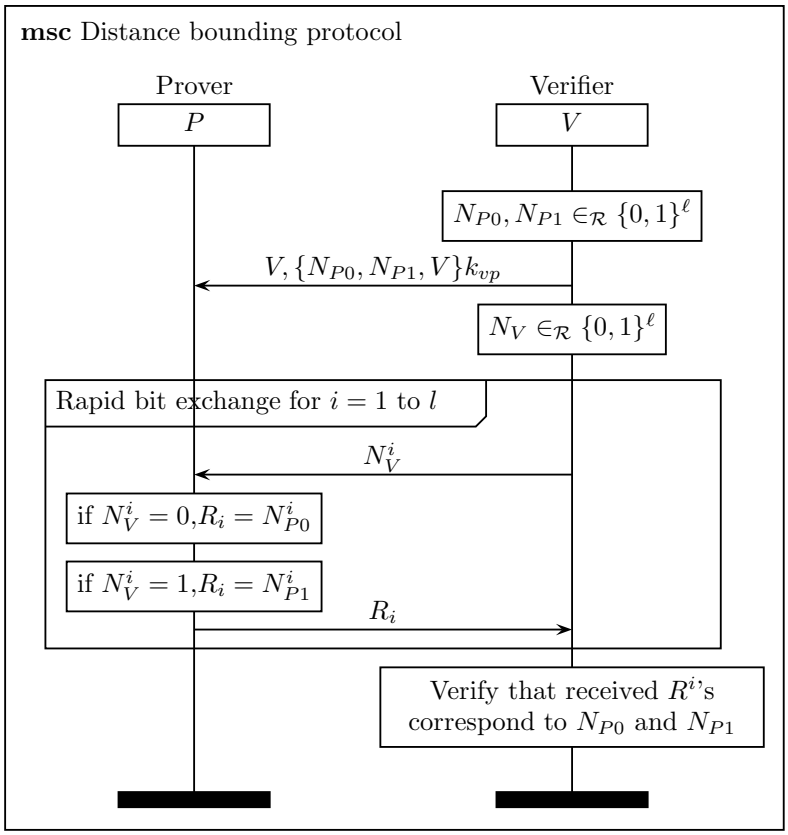

Figure 13: A Distance Bounding Protocol that enables Distance Hijacking on Hancke-Kuhn protocol in multi-protocol environments.

purposes). It might also be that this insecure ranging protocol is simply enabled as a feature for non-critical applications and therefore coexists with the Hancke and Kuhn protocol on the devices (and thus shares the same hardware / distance measurement implementation with the Hancke and Kuhn protocol). This means that no multi-prover distance bounding protocol deployments can be guaranteed to be secure unless additional measures are in place.

In the above example we used an insecure protocol. However, similar attacks are possible using only protocols that are secure in single-protocol environments. We show this on an example of the Hancke-Kuhn distance bounding protocol from [12]. We construct a multi-protocol environment in which the verifier runs the Hancke-Kuhn protocol, and the honest provers support a minor variation of the Hancke-Kuhn protocol that is secure against Distance Hijacking in a single-protocol environment. This protocol, shown in Figure 13, differs from the Hancke and Kuhn protocol in that the prover does not compute the values of registers N_{P0} and N_{P1} but that these are computed by the verifier and sent

(confidentially) to the prover. This protocol modification would make sense if one would, e. g., assume that the prover does not have a good random number generator (e. g., an RFID tag).

A Distance Hijacking attack in this environment works as follows. A dishonest prover P initiates the original Hancke and Kuhn protocol with the verifier V, and derives shared register values with V (for details see Hancke and Kuhn protocol [12]). P then acts as a verifier and initiates the modified Hancke and Kuhn protocol from Figure 13 with the honest prover P'. P then provides the register values to P' as specified in the modified protocol. V and P' then execute a rapid bit exchange and V believes that this exchange was executed by P.

Observe that the attack does not require the two protocols to share the same long-term keys: V verifies the use of the key as prescribed by the Hancke and Kuhn protocol, which was provided by P, and remains unknown to P'. However, the attack strictly requires V and P' to use similar hardware for the fast response phase.

Similarly, a modified version of the Brands and Chaum protocol can be constructed that, if run next to the Hancke and Kuhn protocol, would also enable a Distance Hijacking attack against the Hancke and Kuhn protocol. This phenomenon is similar to the Chosen Protocol attack in cryptographic protocol analysis. We relate the two concepts in Section 8.

Protecting against Distance Hijacking in Multi-Protocol Environments Previously, we proposed countermeasures that prevent Distance Hijacking in single-protocol environments. We now discuss some approaches that can mitigate such attacks in multi-protocol environments.

For multi-protocol environments the obvious solution is to try to ensure that all protocols in an environment use different (incompatible) hardware for their distance measurement phase. This is analoguous to the concepts of *tagging* or *disjoint encryption* for classical cryptographic protocols. Thus, attacks in multi-protocol environments can be prevented by better regulation in distance bounding protocol deployment and construction. Minor application-specific modifications to the distance measurement phase (e. g., including application-specific dummy bits) would already prevent a number of attacks. Similarly, manufacturer-specific or deployment specific hardware modifications would also protect against multi-protocol attacks; this can, however, be expensive.

There are a number of scenarios in which such deployment and regulatory protection measures cannot be used. Application-specific modifications of the distance measurement phase are particularly difficult to implement; given the tight timing constraints in the distance measurement phase, this phase will be processed in hardware. It is also likely that only a few implementations of the distance measurement phase will emerge in the future, limiting available application-specific modifications of this phase. This finally means that most distance bounding protocols will likely use the same implementation of the distance measurement phase.

Accounting for these scenarios, we propose an alternative solution that makes

use of "prover honeypots". Recall that to execute a Distance Hijacking attack, a dishonest prover either needs to be able to successfully claim to have executed a distance measurement phase that was executed by an honest prover, or needs to make an honest prover execute a distance measurement phase using specific bits. The prevention of the false distance measurement claim naturally extends from single- to multi-protocol environments — this type of attack can be prevented by using protocols that are resilient to Distance Hijacking in single-protocol environments. However, as we have shown, protocols that are resilient to Distance Hijacking in single-protocol environments cannot prevent attacks in a multi-protocol environment where an honest prover is made to execute a distance measurement phase using the bits provided by a dishonest prover. We aim to detect such attacks by the use of prover honeypots.

Our solution works as follows. The verifier first sets up a number of virtual or real honeypot provers which are either physical or virtual devices that are placed in the vicinity of the verifier. These honeypot provers are created either by the verifier or by the devices that the verifier trusts and controls. To other provers, honeypot provers claim either their true or false locations/identities, and they support a broad set of ranging and distance bounding protocols. The idea behind this setting is that when a dishonest prover mounts a Distance Hijacking attack, it chooses one of the honeypot provers to abuse in his attack. Besides setting up honeypot provers, the verifier also limits its operation to specific distance bounding protocols: it executes only distance bounding protocols that force a dishonest prover to reveal (most of the bits of) its secret key (that it shares with the verifier) to the honest prover if he wants to execute a Distance Hijacking attack. This is commonly the case for protocols that are resilient against Terrorist Fraud. Thus, if a dishonest prover exploits one of the honeypots in a Distance Hijacking attack, the (majority of the) bits of the key that it shares with the verifier will be revealed to the honeypot prover. For the case in which the prover wants to be certain about the success of Distance Hijacking, all of the bits of his key will be revealed to the honeypot. In order to check if a Distance Hijacking attack was executed, the verifier, after the execution of a distance bounding protocol with a given prover, simply needs to ask his honeypot provers to send him the bits that they used in any recent distance measurement phase. If those bits allow the reconstruction of the (majority of the bits of the) key that the verifier shares with the prover [14], the verifier concludes that the prover attempted to execute a Distance Hijacking attack.

7 Location Hijacking

In this section we generalize Distance Hijacking to *Location Hijacking*. We consider the problem of location verification, or position verification, in which a set of verifiers establishes the location of a prover, even though this prover may act dishonestly, i. e., the prover can pretend to be at another location than he really is. The objective of a location verification protocol is to ensure that the location of the prover is reliably determined. Such protocols often build

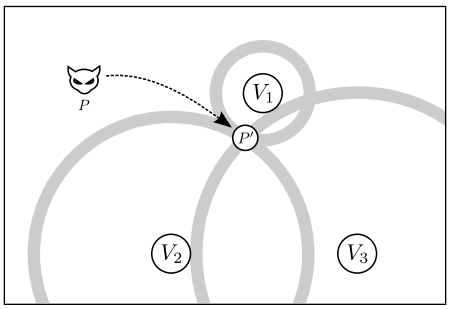

Figure 14: Location Verification and Location Hijacking: P hijacks the location of P', for example by hijacking the distance bounding protocol instances of P' with respect to the verifiers.

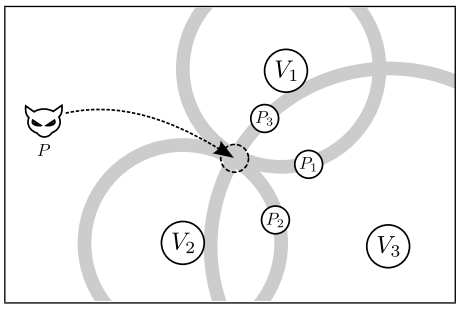

Figure 15: Location Hijacking to empty location: When asked to prove his location to the verifiers, P hijacks P_1's distance for V_1, P_2's distance for V_2, and P_3's distance for V_3. The verifiers conclude that P is at the location indicated by the arrow.

on distance bounding protocols. A prover repeatedly uses a distance bounding protocol to prove his proximity to a set of verifiers. Based on the combined information, the verifiers are able to verify the location of the prover.

This process is depicted in Figure 14. The circles represent the measured distances by the verifiers V_1, V_2, V_3, and they conclude that P' must be located in the intersection. If a dishonest prover P can hijack distance bounding sessions of a party P', he can pretend to be at the location where P' resides, regardless of his actual location. This constitutes a *Location Hijacking* attack: a dishonest prover can hijack the location of P'.

Definition 6. *Location Hijacking attack. A Location Hijacking attack is an attack in which a dishonest prover P exploits one or more honest parties $P_1, \ldots, P_n$ to provide a set of verifiers $V_1, \ldots, V_k$ with false information about the location of P (either absolute or relative to the location of the verifiers).*

The threat of hijacking is magnified in the context of location verification, because multiple distance bounding results are combined. For example, consider the setup in Figure 15, in which three honest provers P_1, P_2, and P_3 are within range of the verifiers. As before, a dishonest prover can perform Distance Hijacking attacks on the distance bounding phases, thereby hijacking the location of P_1, P_2, or P_3 as he chooses. However, he can also combine Distance Hijacking attacks with respect to multiple honest provers: this allows him to make his location appear to be at any intersection of the distances of a set of honest provers. For example, in Figure 15, he can convince the verifiers that he is located at the position indicated by the arrow, by combining the distance bounding phases of the honest provers, even though nobody is present at this location.

Case Study of Location Hijacking To emphasize that Location Hijacking is indeed a relevant problem, even on recent protocols, we give a brief case study of a recent protocol by Chiang et al. [6] that is vulnerable to Location Hijacking. In this protocol a prover sends out a location claim, after which he receives simultaneous challenges from a number of verifiers. The prover aggregates the challenges and broadcasts his response to all verifiers. This many-to-one challenge response then constitutes one round of the underlying distance bounding protocol, which in this case is Brands and Chaum's original suggestion [4]. The authors present a proof that their scheme is optimal in the sense that it achieves the "maximal security" any location verification schemes based solely on time-of-flight can provide.

Despite the proof, this scheme is vulnerable to Location Hijacking (Figure 14). The proof in [6] establishes that a prover must be at the claimed location (within some accuracy) in order to correctly reply to the challenges. The proof however, does not address the authentication of the node at the claimed location but instead leaves that up to the underlying distance bounding protocol. Since the underlying distance bounding protocol is vulnerable to Distance Hijacking, the location verification protocol inherits this vulnerability. In this case it is possible that another distance bounding protocol could be used instead of Brands and Chaum, in order to achieve a secure scheme, but this example shows that even recent location verification schemes with proofs of optimal security, can be vulnerable to Location Hijacking.

8 Related work

Distance bounding for RFID tags Avoine et al. present in [1] a framework for analyzing RFID distance bounding protocols. They give definitions for the three main attack types, and also define *Impersonation Fraud*, in which "a lonely prover purports to be another one" [1, p. 5], i. e., a violation of weak authentication. They consider these four types of attack with respect to black-box and white-box provers, yielding a total of eight security notions. None of their models covers Distance Hijacking attacks.

Dürholz et al. propose in [10] the first computational formal framework for proving properties of RFID distance bounding protocols that are based on shared symmetric keys. Their framework considers an attacker that interacts only with a single prover (the tag) and single verifier (the reader). Consequently, proving that an RFID protocol is secure in their framework does not guarantee the absence of Distance Hijacking attacks.

Formal models for distance bounding Meadows et al. developed a formal methodology to prove properties of distance bounding protocols [18]. Because the methodology is not particularly suited for dealing with dishonest provers, they did not consider scenarios that would allow them to detect Distance Hijacking attacks.

The first two formal approaches for distance bounding protocols that have considered multi-prover scenarios and dishonest provers are Malladi et al. [25] and Basin et al. [3]. Malladi et al. propose a tool-supported framework for analyzing distance bounding protocols, and model a variant of the first signature-based protocol by Brands and Chaum. They analyze this protocol in several scenarios and find an attack that falls into our class. In their "farther adversary" scenario, the attacker is farther from the verifier than the reported distance. This suggests that the "farther adversary" scenario covers both Distance Fraud and Distance Hijacking. However, this observation is not consistent with Malladi et al.'s statement that including the identity in the signature makes the protocol secure in the "farther adversary" scenario. From our analysis it is clear that the resulting protocol will still be vulnerable to Distance Hijacking.

Basin et al. proposed in [3] the basic framework that we have used here as a starting point for our extended model. Basin et al. analyze a family of distance bounding protocols proposed by Meadows et al. in [18] and find an attack that falls into our class of Distance Hijacking attacks, which they refer to as an "impersonation attack". They prove that a concatenation-based version of the protocol is secure in their framework. This protocol is not secure in our extended framework, as the protocol is still vulnerable to a Distance Hijacking attack that uses overshadowing.

Chosen Protocol attack The multi-protocol Distance Hijacking attack described in Section 6 resembles the Chosen Protocol (or Multi-Protocol) attack in cryptographic protocol analysis, which was introduced by Kelsey, Schneier, and Wagner [13]. They describe how, given any secure cryptographic protocol, a second protocol can be constructed ("chosen") that is also secure, but when both are executed in parallel, an attacker can use the second protocol to attack the first. Chosen Protocol attacks are an instance of Multi-Protocol attacks [7]. In a traditional (Dolev-Yao style) setting, Multi-Protocol attacks require that both protocols use the same key infrastructure, in which case many protocols are vulnerable [7]. Ensuring that the protocols use different keys prevents the problem [11], which is often guaranteed in practice. The practical threat of multi-protocol attacks in the Dolev-Yao setting is therefore limited.

In contrast, our multi-protocol Distance Hijacking attacks do not require that keys are shared among protocols. Rather, the distance measurement phase must be regarded as a security primitive, and care must be taken when security primitives are shared among protocols. If not, unexpected interactions can occur, as witnessed by our attacks. In practice, multi-protocol Distance Hijacking poses a more significant threat than Chosen Protocol attacks, because it can be expected that only a few different hardware components for distance measurement will be manufactured, which may be used by a large number of different protocols.

9 Conclusions

In many practical scenarios, including scenarios in which Terrorist Fraud attacks are not a concern, Distance Hijacking attacks can pose a serious threat. Surprisingly, until now, this type of attack was not considered in the analysis of proposed distance bounding protocols.

In fact, our analysis shows that many distance bounding protocols cannot be safely used in scenarios with multiple provers. Fortunately, it seems that adapting the protocols to be resilient against these attacks is possible in single-protocol environments without imposing a significant overhead. Similar observations can be made for location verification protocols with respect to Location Hijacking attacks.

We introduced an extended framework to reason about distance bounding protocols in a symbolic way, which at the same time incorporates bit-level message manipulations by the attacker. The results of this hybrid approach have thus far been promising. The framework enables us to detect more attacks than previous frameworks, including Distance Hijacking attacks, and also allows us to prove the absence of the attacks we found.

We proposed an exhaustive classification of attacks on distance bounding protocols. In this context, we also proposed new names and definitions for the three classical attack types. Our new attack names are less generic and more descriptive than the previous names. Many current works do not attempt to analyse their protocols with respect to all possible threats, and instead provide only a limited analysis of the protocols with respect to some previously defined classes, whose relations and problem coverage are unclear. We hope that our exhaustive classification can contribute to a more systematic analysis of all possible threats against distance bounding protocols.

It is clear that secure functioning in a context with multiple provers is a desirable feature, giving an edge to those protocols that are resilient against Distance Hijacking attacks. It seems therefore prudent to analyze new proposals for distance bounding protocols for their vulnerability to Distance Hijacking.

References

[1] G. Avoine, M. A. Bingöl, S. Kardaş, C. Lauradoux, and B. Martin. A Framework for Analyzing RFID Distance Bounding Protocols. *Journal of Computer Security – Special Issue on RFID System Security*, 2010.

[2] G. Avoine and A. Tchamkerten. An efficient distance bounding RFID authentication protocol: Balancing false-acceptance rate and memory requirement. In *Proceedings of the 12th International Conference on Information Security*, ISC '09, pages 250–261. Springer, 2009.

[3] D. Basin, S. Čapkun, P. Schaller, and B. Schmidt. Let's get physical: Models and methods for real-world security protocols. In *Proceedings of the*

22nd International Conference on Theorem Proving in Higher Order Logics, TPHOLs '09, pages 1–22. Springer, 2009.

[4] S. Brands and D. Chaum. Distance-bounding protocols. In *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *LNCS*, pages 344–359. Springer, 1994.

[5] L. Bussard and W. Bagga. Distance-bounding proof of knowledge to avoid real-time attacks. In *Security and Privacy in the Age of Ubiquitous Computing*, volume 181 of *IFIP Advances in Information and Communication Technology*, pages 223–238. Springer Boston, 2005.

[6] J. T. Chiang, J. J. Haas, and Y.-C. Hu. Secure and precise location verification using distance bounding and simultaneous multilateration. In *Proceedings of the second ACM conference on Wireless network security*, WiSec '09, pages 181–192. ACM, 2009.

[7] C. Cremers. Feasibility of multi-protocol attacks. In *Proc. of The First International Conference on Availability, Reliability and Security (ARES)*, pages 287–294, Vienna, Austria, April 2006. IEEE Computer Society.

[8] Y. Desmedt. Major security problems with the 'unforgeable' (Feige)-Fiat-Shamir proofs of identity and how to overcome them. In *Proceedings of the 6th worldwide congress on computer and communications security and protection (SecuriCom)*, pages 147–159, March 1988.

[9] S. Drimer and S. J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX Security 2007: Proceedings of the 19th USENIX Security Symposium*, 2007.

[10] U. Duerholz, M. Fischlin, M. Kasper, and C. Onete. A formal approach to distance-bounding RFID protocols. In *Information Security Conference (ISC) 2011*, LNCS, 2011. To appear.

[11] J. Guttman and F. Thayer. Protocol independence through disjoint encryption. In *Proc. 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 24–34. IEEE Computer Society, 2000.

[12] G. Hancke and M. Kuhn. An RFID distance bounding protocol. In *Proc. of IEEE/CreatNet SecureComm*, pages 67–73, 2005.

[13] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proc. 5th International Workshop on Security Protocols*, volume 1361 of *LNCS*, pages 91–104. Springer, 1997.

[14] C. Kim, G. Avoine, F. Koeune, F.-X. Standaert, and O. Pereira. The Swiss-Knife RFID distance bounding protocol. In *Information Security and Cryptology ICISC 2008*, volume 5461 of *Lecture Notes in Computer Science*, pages 98–115. Springer, 2009.

[15] C. H. Kim and G. Avoine. RFID distance bounding protocol with mixed challenges to prevent relay attacks. In *Proceedings of the 8th International Conference on Cryptology and Network Security*, CANS '09, pages 119–133. Springer, 2009.

[16] M. Kuhn, H. Luecken, and N. O. Tippenhauer. UWB impulse radio based distance bounding. In *Proceedings of the Workshop on Positioning, Navigation and Communication (WPNC)*, 2010.

[17] G. Lowe. A hierarchy of authentication specifications. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 31–44. IEEE, 1997.

[18] C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, volume 30 of *Advances in Information Security*, pages 279–298. Springer US, 2007.

[19] J. Munilla and A. Peinado. Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels. *Wirel. Commun. Mob. Comput.*, 8:1227–1232, November 2008.

[20] P. Peris-Lopez, J. C. H. Castro, J. M. Estévez-Tapiador, and J. C. A. van der Lubbe. Shedding some light on RFID distance bounding protocols and terrorist attacks. *CoRR*, abs/0906.4618, 2009.

[21] P. Peris-Lopez, J. Hernandez-Castro, J. Tapiador, E. Palomar, and J. van der Lubbe. Cryptographic puzzles and distance-bounding protocols: Practical tools for RFID security. In *RFID, 2010 IEEE International Conference on*, pages 45–52, 2010.

[22] C. Pöpper, N. O. Tippenhauer, B. Danev, and S. Čapkun. Investigation of signal and message manipulations on the wireless channel. In *ESORICS 2011 - 16th European Symposium on Research in Computer Security. Proceedings*, volume 6879 of *LNCS*, pages 40–59. Springer, 2011.

[23] K. B. Rasmussen and S. Čapkun. Realization of RF distance bounding. In *USENIX Security 2010: Proceedings of the 19th USENIX Security Symposium*. USENIX, 2010.

[24] J. Reid, J. M. G. Nieto, T. Tang, and B. Senadji. Detecting relay attacks with timing-based protocols. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, ASIACCS '07, pages 204–213. ACM, 2007.

[25] B. B. S. Malladi and K. Kothapalli. Automatic analysis of distance bounding protocols. In *Foundations of Computer Security*. Affiliated to LICS09, August 2009. Informal proceedings.

[26] B. Schmidt. Isabelle/HOL sources for the extended framework, models, and proofs. Available online at http://www.infsec.ethz.ch/research/software#protoveriphy.

[27] D. Singelée and B. Preneel. Distance bounding in noisy environments. In *Proceedings of the 4th European conference on Security and privacy in ad-hoc and sensor networks*, ESAS'07, pages 101–115. Springer, 2007.

[28] R. Trujillo-Rasua, B. Martin, and G. Avoine. The Poulidor distance-bounding protocol. In *Proceedings of the 6th international conference on Radio frequency identification: security and privacy issues*, RFIDSec'10, pages 239–257. Springer, 2010.

[29] S. Čapkun, L. Buttyán, and J.-P. Hubaux. SECTOR: secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, SASN '03, pages 21–32. ACM, 2003.

[30] S. Čapkun and M. Čagalj. Integrity regions: authentication through presence in wireless networks. In *ACM workshop on Wireless security*, pages 1–10. ACM, 2006.

A Example attack: Brands and Chaum (Schnorr identification)

In [4] several concrete protocols are suggested. None of the proposed protocols is resilient against Distance Hijacking attacks. However, some the attacks are more involved than just re-signing the final message. Below we give a concrete example of such a more involved attack on the protocol from [4, p. 353], which is based on the Schnorr identification scheme.

The Schnorr-based protocol variant is depicted graphically in Figure 16. In the protocol, the public key of the prover is the tuple $(p, q, g, h = g^x \bmod p)$ and the corresponding private key is x.

In the protocol, a prover P chooses a random bit string β and secret random value w. P sends g^w and a commit of β to the verifier V. The verifier chooses his own random bit string α. Next, V performs a rapid bit exchange with P based on α and β. After the rapid bit exchange, P combines α and β into c and computes $r \leftarrow w + cx \bmod p$, i. e., he adds his secret value w to the product of c and his secret key x, modulo q. He sends the resulting r along with the commit opening to V. V also computes c. V then raises the public key of P $(h = g^x)$ to the power of c and multiplies the result with $a = g^w$ and compares the result with g^r. If the values match, the verifier accepts that P is within the measured distance.

In Figure 17 we show a Distance Hijacking attack on the protocol. The honest prover P' starts and tries to prove his distance to the verifier V. The dishonest prover P intercepts the initial message of P' and replaces the value $a_P = g^{w_{P'}}$ by his own $a_P = g^{w_P}$. He sends a_P along with the commit to V. V then performs the rapid bit exchange with P', unaware of the identity mismatch. After this phase, P again intercepts the response of P', effectively replacing the identification computation by his own, while forwarding the "open commit" unchanged.

For the attack, it is necessary that P replaces the value $a_{P'}$ by some value g^{w_P} such that P knows w_P. w_P does not need to be random and may be an arbitrary constant, but it is needed for P to later compute r_P as expected by V.

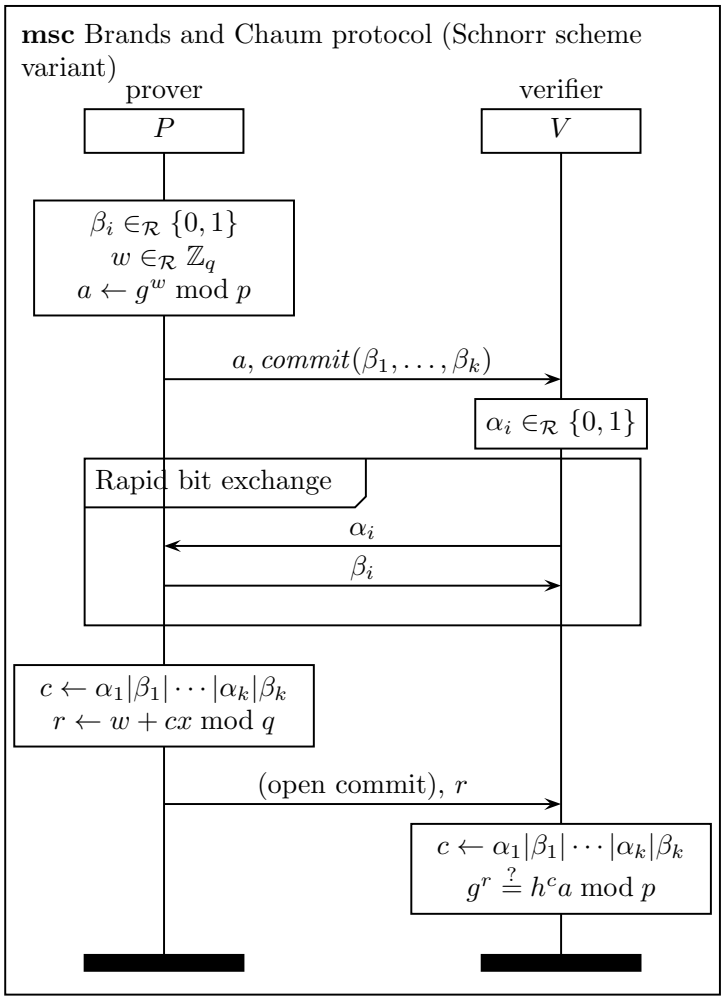

Figure 16: Brands and Chaum protocol based on the Schnorr identification scheme. The public key of the prover is the tuple $(p, q, g, h = g^x \bmod p)$ and the corresponding private key is x.

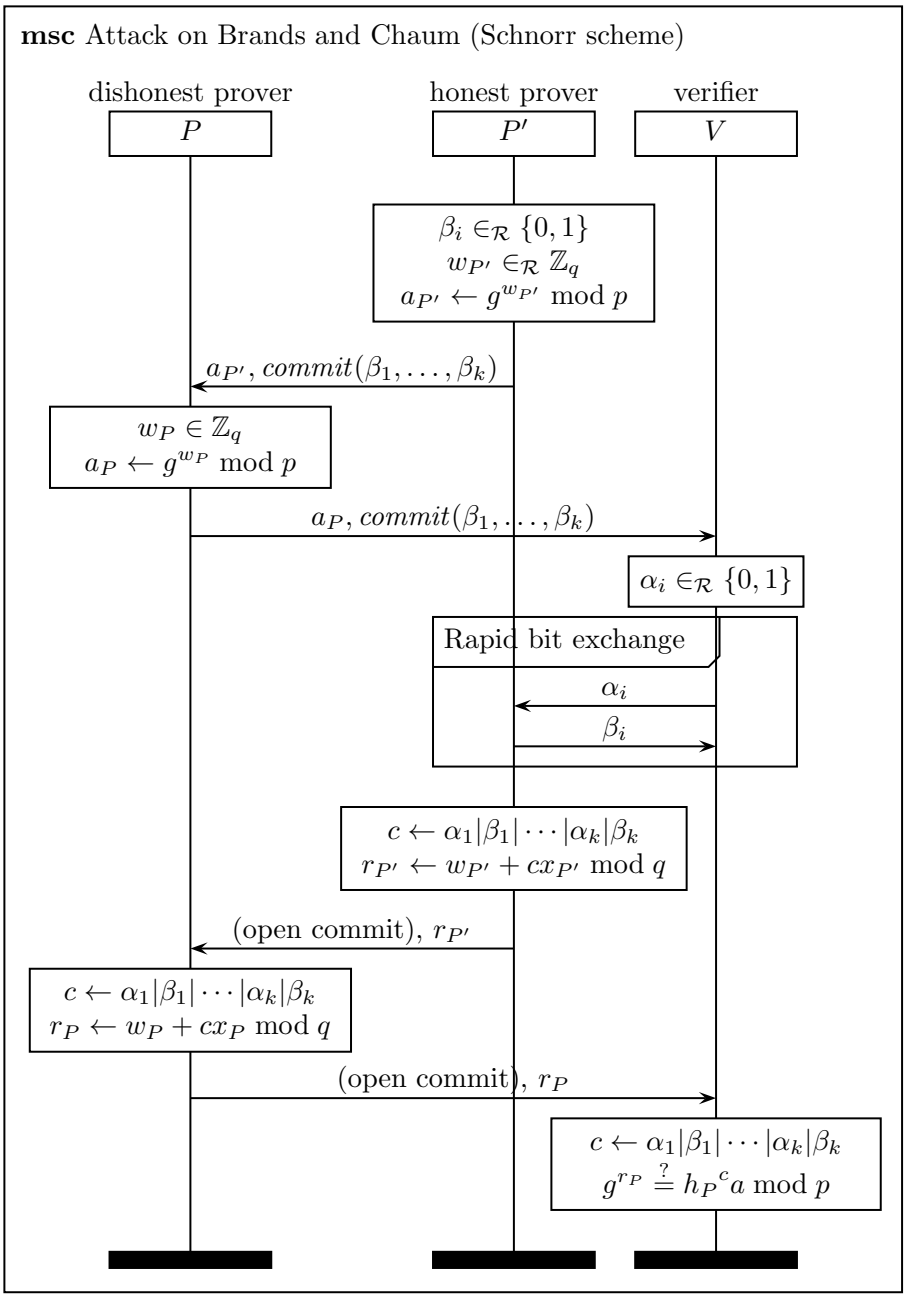

Figure 17: Distance Hijacking attack on Brands and Chaum (Schnorr scheme variant). x_P and $x_{P'}$ are the private keys of P' and P, respectively, and $h_P = g^{x_P}$.