Post-Collusion Security and Distance Bounding

Sjouke Mauw
University of Luxembourg
sjouke.mauw@uni.lu

Zach Smith
University of Luxembourg
zach.smith@uni.lu

Jorge Toro-Pozo
ETH Zurich
jorge.toro@inf.ethz.ch

Rolando Trujillo-Rasua
Deakin University
rolando.trujillo@deakin.edu.au

ABSTRACT

Verification of cryptographic protocols is traditionally built upon the assumption that participants have not revealed their long-term keys. However, in some cases, participants might *collude* to defeat some security goals, without revealing their long-term secrets.

We develop a model based on multiset rewriting to reason about collusion in security protocols. We introduce the notion of *post-collusion security*, which verifies security properties claimed in sessions initiated after collusion occurred. We use post-collusion security to analyse *terrorist fraud* on protocols for securing physical proximity, known as *distance-bounding protocols*. In a terrorist fraud attack, agents collude to falsely prove proximity, whilst no further false proximity proof can be issued without further collusion.

Our definitions and the Tamarin prover are used to develop a modular framework for verification of distance-bounding protocols that accounts for all types of attack from literature. We perform a survey of over 25 protocols, which include industrial protocols such as Mastercard's contactless payment PayPass and NXP's MIFARE Plus with proximity check. For the industrial protocols we confirm attacks, propose fixes, and deliver computer-verifiable security proofs of the repaired versions.

CCS CONCEPTS

• **Security and privacy → Security protocols**; *Formal security models*; *Logic and verification*; *Mobile and wireless security*.

KEYWORDS

security protocols; formal verification; collusion; distance bounding; terrorist fraud

ACM Reference Format:
Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. 2019. Post-Collusion Security and Distance Bounding. In *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19), November 11–15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3319535.3345651

1 INTRODUCTION

Communication protocols are designed with one or more security goals in mind. These goals, such as authentication or confidentiality, must be attainable even in the presence of an adversary attempting to violate them. As such, the capabilities of the adversary is a fundamental consideration in security analysis, continuously undergoing reformations.

In recent years, the adversary model of Dolev and Yao [30] has become an accepted standard. The Dolev-Yao adversary is capable of intercepting, blocking or modifying messages on the communication network, as well as injecting their own messages. Further, the adversary is assumed to be capable of *compromising* protocol participants (a.k.a. users, agents), gaining full control of them for the entire protocol execution.

Most existing verification frameworks make use of the Dolev-Yao adversary, which is sufficient to capture non-trivial attacks, such as Lowe's man-in-the-middle [42] on the Needham-Schroeder protocol [49]. Computer-assisted verification approaches typically consider Dolev-Yao adversaries as well. Such approaches have proven useful in verifying or discovering attacks on real-world, complex protocols such as 5G authentication [11], the TLS 1.3 protocol suite [24], and key-exchange protocols such as Naxos [55].

In some cases, the Dolev-Yao model has been shown to be too coarse-grained. This is because this model assumes that agents can be categorised as being either honest: those who precisely follow their protocol specification; or compromised: those who deviate from their protocol specification as desired by the adversary. The concern lies in accounting for agents who cannot be classified in either group.

For example, *covert adversaries* [1, 18, 33] are agents who are willing to cheat by deviating from the protocol specification, as long as the cheating would not be detected. One might think of an online gaming platform, in which some players secretly cooperate to cheat against other players, whilst avoiding being caught, or else face consequences such as being thrown out of the platform.

Variations of the Dolev-Yao threat model capturing more refined *dishonest* behaviour have been studied [1, 9, 10, 12, 23, 33], which have led to re-thinking the security models to properly account for such fine-grained adversaries. Such models attribute dishonest behaviour to the adversary's compromise capabilities, but in some scenarios such behaviour might not be ruled by the adversary, but rather by the protocol's participants themselves.

For example, a given agent might choose to deviate from the protocol specification, but only if certain guarantees are met in later executions of the protocol. Would a university student willingly, due to certain benefit, lend their campus access card to a university-external friend? The student's decision might be conditional on their assertion that their friend will not be able to later access the campus, after the card has been returned to its owner. Would a user of a video streaming platform utilize a VPN extension to fool geo-location restrictions? The user's decision might be based on whether they are certain that the VPN extension is not malicious and will not cause irreversible harm.

In this paper we refine the traditional Dolev-Yao adversary model in order to capture *collusion*. Collusion refers to any *deviation* of the protocol specification by agents who are not under control of the adversary. Furthermore, we introduce the notion of *post-collusion security*, which refers to security guarantees about claims made in execution sessions initiated after the collusion. Informally, one can interpret the relation of these two notions as follows: post-collusion security allows the potential colluding agents to decide whether colluding is worth it. After all, what the agents gain out of colluding must outweigh the collateral effect that such collusion might have on themselves. On the other hand, a protocol designer might aim to increase the cost of collusion.

A related notion was introduced by Cohn-Gordon *et al.* [23], called *post-compromise security*, that looks at the timeline of the compromise actions and their impact in the security of future protocol sessions. As motivated earlier, collusion differs from compromise in that compromise is an action performed by the adversary in order to exert control over the protocol, whilst collusion represents a deliberate choice of the agent involved.

In post-compromise security, the compromise is ruled by the adversary, regardless of the (future) consequences on the compromised agent. Post-collusion security, instead, allows the agents to base their choice of collusion on post-collusion guarantees. One can think of "not getting caught", in the online gaming example given earlier, as the post-collusion guarantee. In Section 2 we give further technical differences between post-compromise and post-collusion security.

Our notion of post-collusion security finds a straightforward application in *distance-bounding protocols* [13, 15], which are security protocols that aim to guarantee physical proximity. These protocols are used in RFID and NFC technologies, with numerous applications in secure systems such as contactless payment and access control. Post-collusion security allows us to formally analyse a non-trivial type of attack on distance-bounding protocols known as *terrorist fraud* [29]. In this attack, agents collude to falsely prove proximity for one run of the protocol, whereas no further false proximity proofs can be issued without further collusion.

Contributions. The contributions of this paper are:

- We provide a formal symbolic model based on multiset rewriting systems that captures collusion in security protocols, which represents non-compromised agents deviating from their given protocol specification.
- We introduce the notion of post-collusion security, which refers to the validity of security claims made in protocol sessions initiated after the collusion. We provide a concrete formulation of this notion that can be easily implemented in protocol verification tools such as Tamarin [47].
- Our definitions are used to provide a formal description of the sophisticated terrorist fraud on distance-bounding protocols. Further, we develop a Tamarin-based framework for verification of such type of protocols that exhaustively accounts for all classes of attack from literature.
- We conduct a security survey of over 25 protocols, which include industrial protocols based on the ISO/IEC 14443 standard. We propose computer-verified fixes for the vulnerabilities encountered in these protocols.

Organisation. In Section 2 we discuss previous work on modelling alternative adversary models, with a particular focus on distance-bounding protocols and terrorist fraud. We introduce our model in Section 3, which is an extension of the multiset rewriting model employed by the Tamarin tool. In Section 4, we extend the model in order to formalise the concepts of collusion and post-collusion security, and show how these notions apply to authentication analysis. In Section 5, we use post-collusion security to provide a formal definition for terrorist fraud on distance-bounding protocols. We report on our Tamarin-supported framework and verification results in Section 6 and propose fixes for analysed protocols based on the ISO/IEC 14443 standard. We summarise our findings in Section 7.

2 RELATED WORK

In this section we describe some works in which the authors analyse alternative adversary models that modify the Dolev-Yao capabilities. We pay special attention to existing symbolic verification frameworks for distance-bounding protocols, which is the main application field of our findings.

Alternative Adversary Models. In 2010, Basin and Cremers [10] proposed a model in which they formally defined several extensions to the Dolev-Yao adversary. These extensions were used to analyse a variety of protocols against adversaries of varying strength [9]. As a result, they identified new attack vectors in key-exchange protocols such as KEA+ [41], Naxos [40] and the MQV protocol family [38].

In [12] the authors provide a formalism to model and reason about human misbehaviour. A set of rules describe an untrained human, who is willing to perform arbitrary actions but follows a set of guidelines, such as "private keys must be kept secret". The Tamarin tool is used to automatically analyse security protocols involving human errors.

Cohn-Gordon *et al.* introduced post-compromise security in [23], defined as an agent's security guarantees about a communication with their peer, even if their peer has been already compromised. They analysed two types of compromise: weak and total. Weak compromise corresponds to temporary adversarial control of an agent's long-term keys in form of a cryptographic oracle, which outputs the result of a crypto-operation , without revealing the long-term keys. Post-compromise security has been recently used in [22] to analyse group messaging protocols.

The adversary model for post-compromise security is similar to that of post-collusion security in that they both allow for dishonest behavior not conceived by the Dolev-Yao adversary. Yet, they differ in that weak compromise is controlled by the adversary regardless of the compromised agents' will, whilst collusion is the agents' deliberate choice. This choice can be based on whether or not certain post-collusion guarantees are met. Furthermore, Cohn-Gordon *et al.*'s post-compromise security focuses on *stateful* protocols, such as authenticated key-exchange (AKE) and messaging protocols. Our post-collusion security notion can be applied, but is not limited to this type of protocol. In addition, our approach is oriented to symbolic security analysis, whereas theirs uses a computational approach. As a result, our methods can be more smoothly implemented in state-of-the-art verification tools for analysing complex protocols.

Collusion can also be considered in the context of adveraries other than Dolev-Yao, or even with no adversary at all. Tompa and Woll [59] present an attack on Shamir's secret sharing [56], based on the principle of colluding agents. In the domain of multiparty protocols, Hirt and Maurer [35] give a classification of how different agents may deviate from their specification (e.g. 'honest-but-curious' participants, or may collude between each other. Syverson *et al.* [57] build upon an adversary model (named "Machiavelli") which does not directly corrupt agents, but instead manipulates them through an extensive collection of collusion rules. We build upon these papers, by looking at the impact on security *after* collusion occurs, and to make progress towards identifying the key deviations from the protocol specification that will result in "successful" collusion within certain domains.

Distance-Bounding Protocols. A challenge in verifying distance-bounding protocols is precisely the notion of physical distance. Physical distance is associated with location, and so the standard Dolev-Yao model in which the adversary can inject data on the network does not faithfully apply. This is because the Dolev-Yao adversary can inject messages on the network without explicitly annotating the physical location where they have come from.

The first tool-supported framework for symbolic verification of secure distance-bounding was proposed by Basin *et al.* [8, 54]. In this work, the authors *restricted* the Dolev-Yao network capabilities, so that adversarial injection of data is done via a compromised agent. Thus, every message that travels through the network has an origin location associated (the sender's location). The authors use Isabelle/HOL to verify, and find attacks on, a few protocols.

In [44], Mauw *et al.* propose the first verification framework for distance-bounding that does not explicitly require handling the agents' location. Instead, (co-)location is analysed via a causal ordering of the agents' actions. This work provides a considerable improvement over Basin *et al.*'s work, as it reduces code complexity for the protocols' specification as well as verification time and automation level. The authors use the verification tool Tamarin to deliver proofs of the (in)security of a number of protocols.

Both Basin *et al.*'s and Mauw *et al.*'s frameworks allow one to discover traditional distance-bounding attacks such as *mafia fraud* [29], *distance fraud* [28] and *distance hijacking* [26]. These frameworks however do not account for terrorist fraud [29] as non-compromised agents behave precisely as specified by the protocol, ergo not allowing for collusion.

The first formalisation of terrorist fraud within a symbolic model appeared in a recent work by Chothia *et al.* [20]. The authors define collusion actions in the form of a cryptographic oracle similar to weak-compromise in [23]. In their model for terrorist fraud, the adversary queries a distant prover's oracle to obtain the required messages to falsely prove that such prover is co-located with the verifier. In Section 6.2 we discuss differences between Chothia *et al.*'s approach and ours, and show that Chothia *et al.*'s framework delivers incorrect results for some protocols.

3 MODELLING SECURITY PROTOCOLS

This section describes the security model we use throughout this paper. It is based on the multiset rewriting theory employed by the Tamarin verification tool [47, 55]. Protocols are specified as transition rules, and the associated transition system describes the protocol executions. The states of the system are composed of facts. Transition rules model how the protocol participants, as well as the adversary, behave and interact.

3.1 Preliminaries

Notation. Given a set S, we denote by $S^\#$ the set of finite multisets with elements from S, and by S^* the set of finite sequences of elements from S. The power set of S is denoted by $\mathcal{P}(S)$. For any operation on sets, we use the superscript $^\#$ to refer to the corresponding operation on multisets.

Given a (multi)set S and a sequence $s \in S^*$, $|s|$ denotes the length of s, s_i the i-th element of s with $1 \leq i \leq |s|$, and λ the empty sequence. We write s indistinctly as $[s_1, \ldots, s_n]$ or $s_1 \cdots s_n$ (the choice depends on presentation). The concatenation of the sequences s and s' is denoted by $s \cdot s'$. We use $set(s)$ and $multiset(s)$ to denote the set and multiset of elements from s, respectively. Given $a \in S$ and $s \in S^*$, we write $a \in s$ for $\exists i \in \{1, \ldots, |s|\}.\ a = s_i$ and $a \notin s$ for the opposite.

Cryptographic Messages. To model cryptographic messages, we use an order-sorted term algebra $(\mathcal{S}, \leq, \mathcal{T}_\Sigma(\mathcal{V}))$ where $\mathcal{S}$ is a set of sorts, $\leq$ a partial order on $\mathcal{S}$, Σ is a signature, and $\mathcal{V}$ is a countably infinite set of variables. We consider three sorts: $msg, fresh, pub \in \mathcal{S}$, where $fresh \leq msg$ and $pub \leq msg$. That is, msg is the super sort of two incomparable sub-sorts $fresh$ and pub, denoting fresh and public names, respectively. We use $\mathcal{V}_s \subseteq \mathcal{V}$ to denote the set of term variables of sort s and write $x\colon s$ to indicate that x is a variable of sort s.

Each function symbol $f \in \Sigma$ has a type $(w, s) \in \mathcal{S}^* \times \mathcal{S}$, where w is the *arity* and s the *sort*. If w is the empty sequence λ, then f denotes a constant of sort s. We use $\Sigma_{w,s} \subseteq \Sigma$ to denote the family of function symbols of type (w, s). Special function families are $\Sigma_{\lambda,fresh}$ and $\Sigma_{\lambda,pub}$, denoting fresh names (a.k.a. nonces) and public names, respectively. Public names include constants (often written in between quotations, e.g. '`hello`'), and agents' names.

We provide next a list of reserved function symbols:

- $\langle\ ,\ \rangle \in \Sigma_{msg\times msg, msg}$ to pair two terms.
- fst, snd $\in \Sigma_{msg, msg}$ to extract the first and second term from a pair, respectively.
- senc, sdec $\in \Sigma_{msg\times msg, msg}$ for symmetric encryption and decryption, respectively. The second argument is the key.
- aenc, adec $\in \Sigma_{msg\times msg, msg}$ for asymmetric encryption and decryption, respectively.
- pk $\in \Sigma_{msg, msg}$ to indicate the asymmetric public key of the argument.
- sign $\in \Sigma_{msg\times msg, msg}$ and verify $\in \Sigma_{msg\times msg\times msg, pub}$ to create and verify signatures, respectively.

The semantics of the function symbols above is formalised by an equational theory E, which is in turn defined by the following set of equations over Σ, where true $\in \Sigma_{\lambda,msg}$:

$$\begin{aligned}\{&\text{fst}(\langle x,y\rangle)=x,\ \text{snd}(\langle x,y\rangle)=y,\\ &\text{sdec}(\text{senc}(x,y),y)=x,\\ &\text{adec}(\text{aenc}(x,\text{pk}(y)),y)=x,\\ &\text{verify}(\text{sign}(x,y),x,\text{pk}(y))=\text{true}\}.\end{aligned}$$

We use $t =_E t'$ to indicate that terms t and t' are equal modulo E. Terms in our term algebra without free variables are called *ground* terms. A *substitution* is a well-sorted function $\sigma\colon \mathcal{V} \to \mathcal{T}_\Sigma(\mathcal{V})$, i.e. $(\sigma(x) = y \wedge x\colon s) \implies y\colon s$, from variables to terms. We use $t\sigma$ to denote the application of the substitution σ to the term t.

Multiset Rewriting System. We model the execution of a protocol as a labelled transition system. A state in the system is a multiset of *facts*, and a fact is a term of the form $F(t_1,\ldots,t_n)$ where F is a symbol from an unsorted signature Γ and $t_1,\ldots,t_n$ are terms in $\mathcal{T}_\Sigma(\mathcal{V})$. For $n \geq 0$ we denote by $\Gamma^n \subseteq \Gamma$ the set of fact symbols with n arguments. The application of a substitution function σ to a fact $F(t_1,\ldots,t_n)$, denoted $F(t_1,\ldots,t_n)\sigma$, results in the fact $F(t_1\sigma,\ldots,t_n\sigma)$. The set of all facts is denoted $\mathcal{F}$ and the set $\mathcal{G} \subseteq \mathcal{F}$ denotes the set of *ground facts*, which are facts with only ground terms as arguments.

We extend equality modulo E from terms to facts as follows: $F(t_1,\ldots,t_n) =_E G(t'_1,\ldots,t'_m)$ if and only if $F = G$, and $n = m$, and $t_i =_E t'_i$ for all $i \in \{1,\ldots,n\}$. Substitution and equality modulo E are extended to sequences of facts in the trivial way.

A fact symbol is either *linear* or *persistent*. Linear fact symbols model resources that are exhaustible, such as a message sent to the network. Persistent fact symbols model inexhaustible resources, such as the adversary knowledge or long-term encryption keys.

We reserve the linear fact symbols $\mathsf{In}, \mathsf{Out}, \mathsf{Fr} \in \Gamma^1$. The facts $\mathsf{In}(m)$ and $\mathsf{Out}(m)$ denote the reception and sending of m, respectively. $\mathsf{Fr}(m)$ indicates that m is a fresh name.

The persistent fact symbols $\mathsf{K} \in \Gamma^1$, $\mathsf{Ltk} \in \Gamma^2$, $\mathsf{Shk} \in \Gamma^3$ and $\mathsf{Compromise} \in \Gamma^1$ are also reserved. $\mathsf{K}(m)$ indicates that the message m is known to the adversary. Facts with symbols Shk and Ltk are used to associate agents to their *long-term* cryptographic keys. $\mathsf{Shk}(A,B,k)$ indicates that k is the long-term symmetric key shared by A and B, and $\mathsf{Ltk}(A,sk)$ indicates that A holds the long-term asymmetric private key sk. We say that an agent A is compromised if the agent has revealed at least one of their long-term keys; and we use the fact $\mathsf{Compromise}(A)$ to indicate so.

Given a sequence of facts $s \in \mathcal{F}^*$, we write $linear(s)$ and $persist(s)$ to denote the *multiset* of linear facts from s, and the *set* of persistent facts from s, respectively.

The execution of a protocol starts with the empty multiset of facts, and evolves through *multiset rewriting rules.* A multiset rewriting rule is a tuple (p,a,c), written as $[p]\overset{a}{\longrightarrow}[c]$, where p, a and c are sequences of facts called the *premises*, the *actions*, and the *conclusions* of the rule, respectively. Each term in a multiset rewriting rule is assumed to be of sort *msg*, unless otherwise indicated.

A *ground instance* of a rule $r := [p]\overset{a}{\longrightarrow}[c]$ is obtained via application of a substitution function σ to result in $r\sigma := [p\sigma]\overset{a\sigma}{\longrightarrow}[c\sigma]$ where $p\sigma$, $a\sigma$ and $c\sigma$ consist of ground facts only. Given a set of rules R, we denote $ginsts(R)$ the set of ground instances of the rules from R. We write $g \in_E G$, where g is a (possibly ground) rule and G is a set of (possibly ground) rules, to indicate that $\exists g' \in G.\ g =_E g'$.

A set R of multiset rewriting rules defines a *multiset rewriting system*: an LTS whose set of states is $\mathcal{G}^\sharp$ and whose transition relation $\to_R \subseteq \mathcal{G}^\sharp \times \mathcal{P}(\mathcal{G}) \times \mathcal{G}^\sharp$ is defined by:

$$\begin{aligned}&S \xrightarrow{l}_R S' \iff\\ &\quad \exists(p,a,c) \in_E ginsts(R).\\ &\qquad l = set(a) \wedge linear(p) \subseteq^\sharp S \wedge persist(p) \subseteq set(S) \wedge\\ &\qquad S' = \left(S \setminus^\sharp linear(p)\right) \cup^\sharp multiset(c). \qquad (1)\end{aligned}$$

A transition is performed by applying a ground instance of a transition rule. The rule is applicable if the current system state contains all facts in the premises of the rule. The rule application removes the linear facts from the state, keeps the persistent facts, and adds the facts from the conclusions.

An *execution* of R is a finite sequence $[S_0,l_1,S_1,\ldots,l_n,S_n]$ alternating states and labels such that:

- $S_0 = \emptyset^\sharp$,
- $S_{i-1} \xrightarrow{l_i}_R S_i$ for $1 \leq i \leq n$, and
- if $S_{i+1} \setminus^\sharp S_i = \{\mathsf{Fr}(x)\}^\sharp$ for some i and x, then $j \neq i$ does not exist such that $S_{j+1} \setminus^\sharp S_j = \{\mathsf{Fr}(x)\}^\sharp$.

The third condition guarantees that fresh names are generated once. The set of all executions of R is denoted $[\![R]\!]$.

3.2 Protocol Specification

A protocol is specified as a set of multiset rewriting rules, called *protocol rules*, with the following restrictions: (1) fresh names and K facts are not used, (2) In and Fr facts do not occur in the conclusions, and (3) every variable occurring in the actions or conclusions either occurs in the premises or is of sort *pub*. The universe of all rules that satisfy these conditions is denoted $\mathcal{R}$.

Example 1 (The *Toy* protocol). *Figure 1 shows a message sequence chart (MSC) [25] of the Toy protocol, an example protocol which we will use for illustration throughout the paper. The initiator I creates a nonce*[1] *ni, and sends it to the responder agent R, encrypted with their shared long-term symmetric key. Upon reception, R decrypts the received message to learn ni. Then, R creates his own nonce nr, encrypts it using the nonce ni as a key, and sends that encrypted message to I. Upon reception of* senc(nr,ni)*, I learns nr and sends back to R a hash of nr. Such a value allows R to be convinced that I has executed the protocol with R and agrees on the nonces nr and ni. The protocol rules are depicted in Figure 2.*

The specification of the *Toy* protocol uses fact symbols that are reserved, such as Shk. Indeed, we assume that all protocol specifications use reserved fact symbols with the intended meaning. The remaining facts are used to enrich execution traces with information that will be later used to analyze trace properties.

For example, $\mathsf{RState1}(I,R,ni,nr)$ appears in the conclusion of R1 and in the premises of R2, allowing to establish an order between R1 and R2. The facts of the form $\mathsf{Start}(x)$ and $\mathsf{End}(x)$ denote the start and the end of a protocol run by an agent, respectively. The term

[1]We will indistinctly use "nonce" and "fresh name", though they mean the same thing: a number generated once.

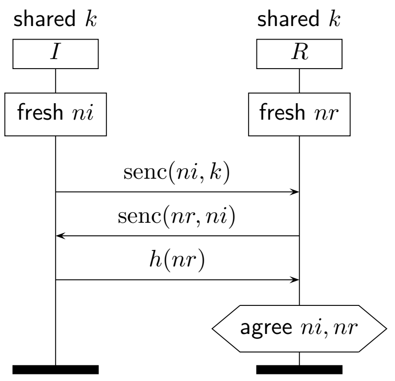


Figure 1: The *Toy* protocol.

$$\mathtt{KeyGen} := [\mathsf{Fr}(k)] \longrightarrow [\mathsf{Shk}(I, R, k)]$$

$$\mathtt{ShkCompI} := [\mathsf{Shk}(I, R, k)] \xrightarrow{\mathsf{Compromise}(I)} [\mathsf{Out}(k)]$$

$$\mathtt{ShkCompR} := [\mathsf{Shk}(I, R, k)] \xrightarrow{\mathsf{Compromise}(R)} [\mathsf{Out}(k)]$$

$$\mathtt{I1} := [\mathsf{Fr}(ni),\ \mathsf{Shk}(I, R, k)] \xrightarrow{\mathsf{Start}(ni)} \begin{bmatrix} \mathsf{Out}(\mathrm{senc}(ni, k)), \\ \mathsf{IState1}(I, R, ni) \end{bmatrix}$$

$$\mathtt{R1} := \begin{bmatrix} \mathsf{Fr}(nr),\ \mathsf{In}(\mathrm{senc}(ni, k)), \\ \mathsf{Shk}(I, R, k) \end{bmatrix} \xrightarrow{\mathsf{Start}(nr)} \begin{bmatrix} \mathsf{Out}(\mathrm{senc}(nr, ni)), \\ \mathsf{RState1}(I, R, ni, nr) \end{bmatrix}$$

$$\mathtt{I2} := \begin{bmatrix} \mathsf{IState1}(I, R, ni), \\ \mathsf{In}(\mathrm{senc}(nr, ni)) \end{bmatrix} \xrightarrow[\mathsf{End}(ni)]{\mathsf{Running}(I,R,\langle \text{'I'}, \text{'R'}, ni, nr\rangle),} [\mathsf{Out}\,(h(nr))]$$

$$\mathtt{R2} := \begin{bmatrix} \mathsf{RState1}(I, R, ni, nr), \\ \mathsf{In}\,(h(nr)) \end{bmatrix} \xrightarrow[\mathsf{End}(nr)]{\mathsf{Commit}(R,I,\langle \text{'I'}, \text{'R'}, ni, nr\rangle),} [\ ]$$

Figure 2: Specification rules of the *Toy* protocol.

x denotes the run identifier. We delay the discussion of the fact symbols Commit and Running until the introduction of security properties in Section 3.4.

For the remainder of this article, the fact symbols Start and End are reserved to mark the start and end of the protocol execution. Also we assume that the protocol specification is consistent with the usage of Start and End. In particular, we assume that all End facts are reachable from the empty state.

3.3 Execution and Adversary Model

Given that protocol rules cannot generate fresh names, i.e. protocol rules are not allowed to use Fr facts in their conclusion, we add a special rule Fresh, independent of the protocol specification, to model generation of fresh names:

$$\mathtt{Fresh} := [\ ] \mapsto [\mathsf{Fr}(x)] \text{ where } x\colon \mathit{fresh}.$$

To model the adversary's actions we use the standard Dolev-Yao network adversary, modelled by the rules depicted in Figure 3, which we will explain in the next paragraph. We remark that the compromise capabilities of the adversary are part of the protocol

$$\mathtt{Learn} := [\mathsf{Out}(x)] \mapsto [\mathsf{K}(x)]$$

$$\mathtt{Inject} := [\mathsf{K}(x)] \xrightarrow{\mathsf{K}(x)} [\mathsf{In}(x)]$$

$$\mathtt{AdvFresh} := [\mathsf{Fr}(x)] \mapsto [\mathsf{K}(x)]$$

$$\mathtt{Public} := [\ ] \mapsto [\mathsf{K}(x)] \text{ where } x\colon \mathit{pub}$$

$$\mathtt{Funct} := [\mathsf{K}(x_1), \ldots, \mathsf{K}(x_n)] \mapsto [\mathsf{K}\,(f(x_1, \ldots, x_n))]$$

Figure 3: Dolev-Yao rules.

specification (e.g. KeyCompI and KeyCompR in the *Toy* protocol). We will extend this model with collusion actions in the next section.

The rules Learn and Inject model the adversary's ability to learn messages being sent and to inject any of their known messages, respectively. The rule AdvFresh declares that the adversary can generate their own fresh names. The rule Public states that the adversary knows all public messages and the rule Funct indicates that the adversary can evaluate any function, provided that they know the inputs.

We denote by $\mathcal{I}$ the set of intruder rules in Figure 3 together with the rule Fresh, which will form part of every protocol specification. Hence, the set of all executions of a set *Proto* of protocol rules is $[\![\mathit{Proto} \cup \mathcal{I}]\!]$. Moreover, given an execution $[S_0, l_1, S_1, \ldots, l_n, S_n]$ of *Proto*, the sequence $l_1 \cdots l_n$ is called the *trace*. The set of all traces of *Proto* is denoted *Traces*(*Proto*).

3.4 Security Properties

Security properties are verified on execution traces. Certain facts on the traces indicate a security *claim*, e.g. the Commit fact in the *Toy* protocol. A security claim denotes a belief (traditionally of an agent) about the protocol execution that led to the claim. Formally, we define a security property φ as a relation on traces and integer numbers such that $\varphi(t_1 \cdots t_n, i)$ means that security claims in t_i are valid. Recall that a trace is a sequence of labels, which in turn are sets of ground facts.

For illustration purposes, let us instantiate φ with the authentication property *non-injective agreement*, as defined by Lowe in [43]. Following Lowe's notation, we use the fact symbols Running and Commit as markers in the traces to indicate those execution steps where agreement is expected to be satisfied, e.g. as used by rules I2 and R2 in Figure 2.

Non-injective agreement on a message m is guaranteed to an agent A, if whenever A completes a run apparently with B, denoted by the claim Commit(A, B, m), then B has previously performing a run apparently with A and they both agree on m, denoted by the fact Running(A, B, m):

$$\begin{aligned} &\mathit{ni\_agreement}(t, i) \iff \\ &\quad \forall A, B, m.\ \mathsf{Commit}(A, B, m) \in t_i \implies \\ &\quad\quad \big(\exists j.\ \mathsf{Running}(B, A, m) \in t_j\big) \ \vee \\ &\quad\quad \big(\exists j.\ \mathsf{Compromise}(A) \in t_j \vee \mathsf{Compromise}(B) \in t_j\big)\,. \end{aligned} \tag{2}$$

On the one hand, the prefix-closure of traces imposes an implicit order $j < i$ in Equation 2, which is suggested by the word "previously" in the description of the property. On the other hand, the last line of Equation 2 indicates that the property is conditional on A and B not being compromised.

Definition 1 (Security). *A set Proto of protocol rules* satisfies *a security property φ, denoted Proto $\models \varphi$, if:*

$$\forall t \in \mathit{Traces}(\mathit{Proto}), i \in \{1, \ldots, |t|\}.\ \varphi(t, i).$$

The *Toy* protocol satisfies[2] non-injective agreement, i.e.

$$\mathit{Toy} \models \mathit{ni\_agreement}.$$

We write $\mathit{Proto} \not\models \varphi$ to indicate that $\mathit{Proto} \models \varphi$ does not hold.

4 COLLUSION

The security model introduced in the previous section can be used to model and verify standard security properties, e.g. secrecy and agreement. This section is dedicated to providing a formal description of the notions of *collusion*, which is an extension to the adversary model, and *post-collusion security*, which is a security model under the extended adversary.

More precisely, in Section 4.1 we extend the adversary model with collusion rules, which express ways in which non-compromised agents can deviate from the protocol specification. In an illustrative example, we show how such extension invalidates, under the traditional security verification model (Definition 1), the agreement-based authentication property given earlier (Section 3.4). Later, in Section 4.2 we provide the formulation of post-collusion security.

4.1 Collusion Rules

In the traditional Dolev-Yao compromise model, agents are assumed to be either *compromised* (a.k.a. corrupt, dishonest) or *non-compromised* (a.k.a. honest). Non-compromised agents follow precisely the protocol specification, whilst compromised agents deviate from it as pleased by the adversary.

We refine the traditional Dolev-Yao compromise model so that agents can *collude* in order to provide false proof to their communication partners of a certain claim's validity. Collusion refers to non-compromised agents' deviation from their protocol specification. The basic deviation consists of leakage of session data, cryptographic oracles, reuse of nonces, or state reveals.

For example, assume *Alice* is running an authentication protocol (supposedly) with *Bob*. Consider also a third party *Charlie* who, in cooperation with *Bob*, impersonates *Bob* when communicating with *Alice*. *Bob* could trivially achieve this by giving all his secret keys to *Charlie*. But, does *Bob* really have to do so in order to deceive *Alice*? Not necessarily. Indeed, *Bob* can provide *Charlie* (possibly in advance) with all the messages that *Charlie* needs to successfully complete a protocol session with *Alice*, posing as *Bob*. Such aid by *Bob* is what we call collusion, and we call *Bob* a colluding agent.

Collusion is modelled by extending the protocol specification. For example, in the *Toy* protocol, I might collude with a compromised agent, say *Eve*, by leaking *ni*. This can be modelled with the rule:

$$\mathtt{Leak\_ni} := \left[\mathsf{IState1}(I, R, ni)\right] \xrightarrow{\mathsf{Collusion}(\,)} \begin{bmatrix}\mathsf{IState1}(I, R, ni), \\ \mathsf{Out}(ni)\end{bmatrix}. \quad (3)$$

Such rule leads to the following statement:

$$\mathit{Toy} \cup \{\mathtt{Leak\_ni}\} \not\models \mathit{ni\_agreement}. \quad (4)$$

Another example of a deviation is an encryption oracle, which can be modelled as follows:

$$\mathtt{EncOracle} := \begin{bmatrix}\mathsf{In}(m), \\ \mathsf{Shk}(I, R, k)\end{bmatrix} \xrightarrow{\mathsf{Collusion}(\,)} \left[\mathsf{Out}(\mathrm{senc}(m, k))\right]. \quad (5)$$

The rule `EncOracle` models the reveal of the encryption of a message (possibly adversary-chosen) with a shared symmetric key. This leads to:

$$\mathit{Toy} \cup \{\mathtt{EncOracle}\} \not\models \mathit{ni\_agreement}. \quad (6)$$

The rules that extend the protocol specification to model collusion are called *collusion rules*. By convention, and also to syntactically distinguish legitimate protocol rules from collusion rules, we will assume that all collusion rules have an action fact of the form Collusion(). We denote by $\mathcal{C} \subseteq \mathcal{R}$ the universe of all collusion rules. We restrict the set of collusion rules by requiring them to not prevent agents from completing legitimate protocol runs.

Definition 2 (Valid Extension). *Let Proto $\subseteq \mathcal{R} \backslash \mathcal{C}$ be a protocol and $C \subseteq \mathcal{C}$ be a set of collusion rules, we say that $\mathit{Proto}' = \mathit{Proto} \cup C$ is a* valid extension *of Proto if:*

$$\begin{aligned}&\forall \alpha \in \mathit{Traces}(\mathit{Proto}'), i, x. \\ &\quad (\mathsf{Start}(x) \in \alpha_i \wedge \nexists j.\ \mathsf{End}(x) \in \alpha_j) \implies \\ &\qquad \exists \beta.\ \alpha \cdot \beta \in \mathit{Traces}(\mathit{Proto}') \wedge \mathsf{End}(x) \in \beta_{|\beta|}.\end{aligned}$$

Definition 2 states that collusion rules do not create points of no-return during execution. That is to say, agents must always be able to complete their runs even if they have colluded. For example, $\mathit{Toy} \cup \{\mathtt{Leak\_ni}\}$ is a valid extension of *Toy* because, even if I leaks *ni* in some execution, I can still continue with the intended protocol execution.

An example of a rule that leads to a non-valid extension of *Toy* is the following:

$$\mathtt{NonValidRule} := \left[\mathsf{IState1}(I, R, ni)\right] \xrightarrow{\mathsf{Collusion}()} [\ ].$$

$\mathit{Toy} \cup \{\mathtt{NonValidRule}\}$ is not a valid extension of *Toy* because if this rule is applied, it will consume the fact $\mathsf{IState1}(I, R, ni)$ from the system state, and so I will never be able to continue with the current run (identified by *ni*).

Other than this requirement of not preventing termination, we place no other restrictions on collusion rules. Besides leakage rules or function oracles (as demonstrated), we also allow for more esoteric deviations, such as re-use of fresh values, or passing of messages between multiple colluding protocol participants (and not an adversary or fully compromised agent).

4.2 Post-Collusion Security

In this section we introduce the notion of post-collusion security. We informally define it as follows.

Definition 3 (Informal). *Post-collusion security is the guarantee of security claims made in sessions initiated* after *collusion occurs.*

The remainder of this section is intended to formalise the above informal definition of post-collusion security. Moreover, we will

[2] All Tamarin models and proofs used for this paper can be found in our repository https://github.com/jorgetp/dbverify.

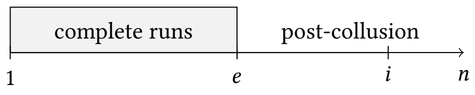


Figure 4: A trace $t = t_1 \cdots t_e \cdots t_i \cdots t_n$ can be broken down into a pre-collusion trace consisting of completed runs (e.g. before e), and a second subtrace containing post-collusion claims (e.g. a claim made in t_i).

use the *Toy* protocol and the agreement property from the previous sections to illustrate our definitions and intuitions.

To identify claims made in sessions initiated after the collusion, which we call *post-collusion claims*, we must make sure that all sessions before or while the (last) collusion occurred are complete. The reason for this is that an agent who makes a security claim cannot always decide whether their communication partner is still acting on a run initiated before or during the collusion. That is, a claim by *Alice* about her communication with *Bob* is a post-collusion claim if both *Alice* and *Bob* have completed their runs that started before or while *Bob* performed the collusion action(s). That way, we make sure that *Alice* makes her claim in a session initiated after *Bob*'s collusion action.

Consider a trace $t = t_1 \cdots t_e \cdots t_i \cdots t_n$, and an index e such that all collusion actions (if any) occurred before e. If all runs initiated before e were completed before e too, then we call the security claims made after e post-collusion claims. See Figure 4 for a graphical representation. Note that every claim that occurs after a post-collusion claim is also a post-collusion claim.

Below, in Definition 4 we formulate post-collusion security, in which we use the following helper predicates on sequences of sets of ground facts:

$$complete(l) \iff \forall i, x.\ (\mathsf{Start}(x) \in l_i \implies \exists j.\ \mathsf{End}(x) \in l_j),$$
$$nocollusion(l) \iff \nexists j.\ \mathsf{Collusion}() \in l_j.$$

In words, $complete(l)$ holds if all runs initiated in l are also completed in l; $nocollusion(l)$ means that no collusion actions occurred in l. We note that the $complete()$ predicate gives a strong divide between the complete runs and the post-collusion runs. However, we assert that for interleaved traces (i.e. those in which there is always an active session), there is an equivalent trace which satisfies the predicate. Intuitively, sessions between unrelated agents have no causal dependence and so can be reordered. This leaves only series of sessions by the same agents. In the case that an agent may be participating in multiple sessions simultaneously, we must require that all of them are finished before we can make post-collusion claims. This is because we cannot guarantee that a collusion action taken in one session will necessarily only lead to attacks in that session - for example, an agent may collude by acting as a function oracle that can be used in any one of their active sessions.

Definition 4 (Post-collusion Security). *Given a protocol Proto, a valid extension Proto′ of Proto, and a security property φ, we say that Proto′* is post-collusion secure with respect to φ, *denoted Proto′ $\models^\star \varphi$, if:*

$$\forall t \in Traces(Proto'), e \in \{1, \dots, |t|\}.$$
$$(complete(t_1 \cdots t_e) \wedge nocollusion(t_{e+1} \cdots t_{|t|}))$$
$$\implies \forall i > e.\ \varphi(t, i). \tag{7}$$

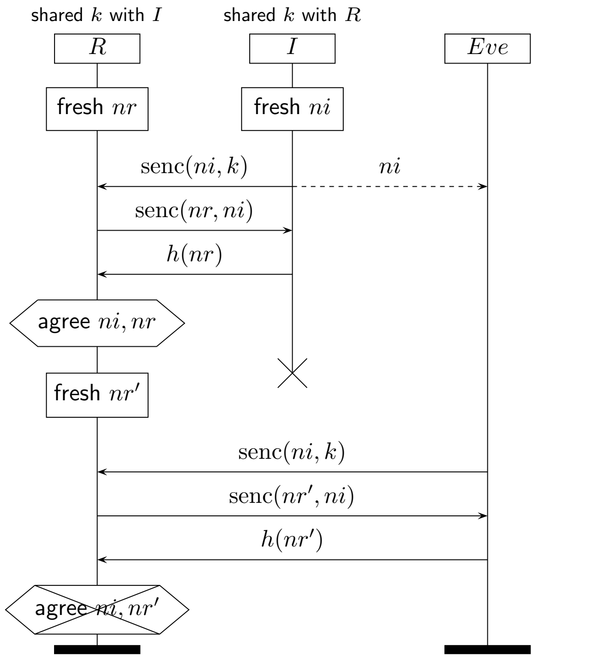


Figure 5: An MSC showing that the *Toy* protocol with collusion, represented by the dashed arrow, is not post-collusion secure with respect to non-injective agreement.

We write *Proto′* $\not\models^\star \varphi$ to indicate that *Proto′* $\models^\star \varphi$ does not hold. As Figure 5 shows, *Toy* $\cup$ {`Leak_ni`} is *not* post-collusion secure with respect to non-injective agreement, i.e.

$$Toy \cup \{\mathtt{Leak\_ni}\} \not\models^\star ni\_agreement. \tag{8}$$

The attack works with two consecutive sessions, in which a compromised agent *Eve* can re-use the messages senc(ni, k) and ni from the first session to impersonate I in the second session. Observe that the second claim is a post-collusion claim, as the first session is complete and no collusion occurred in the second session.

The impact of post-collusion security can depend on the circumstances in which a given protocol is deployed. We see from the *Toy* protocol that the effects of collusion can cause an irreversible change to the truth value of future authentication claims. Thus, a legitimate agent playing the initiator role would not want to collude with a "friend" by giving them their nonce ni, as this would lead to impersonation. On the contrary, suppose a given protocol is post-collusion secure with respect to a desirable authentication property. Then, an agent can issue their one-time keys to their friends if desired, confident that these friends will not be able to re-use this information for later authentication.

5 DISTANCE BOUNDING AND TERRORIST FRAUD

In this section we use post-collusion security to develop a symbolic formulation of (resistance to) terrorist fraud in distance-bounding protocols. First, in Section 5.1, we describe how to model such protocols by using the multiset rewriting model from Section 3. We pay particular attention to restrictions to the Dolev-Yao model

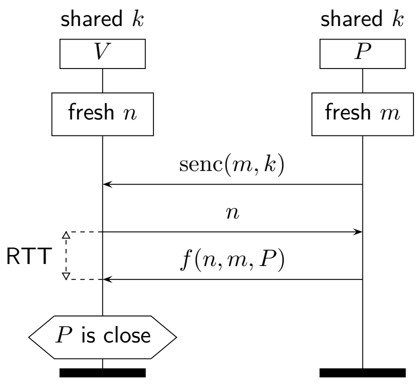


Figure 6: The *DBToy* protocol.

that are necessary to model physical limitations of the communication channel. Later on, in Section 5.2, we formulate the *secure distance-bounding* property proposed in [44] to verify this type of protocols. Finally, in Section 5.3, we provide a symbolic formulation of resistance to terrorist fraud.

5.1 Modelling Distance-Bounding Protocols

Distance-bounding protocols are security protocols that aim to guarantee physical proximity between the participants. These protocols determine proximity by checking that the round trip times (RTT) of a number of challenge/response cycles are below a certain threshold. The phase of the protocol where the RTTs are measured is called the *fast phase*. See next a running example.

Example 2 (The *DBToy* Protocol). *Figure 6 depicts the DBToy protocol, which works as follows. The prover P encrypts a fresh name m with the shared key between P and the verifier V. Then P sends the encrypted message to V. Hence, the fast phase starts with V sending the fresh name n as the challenge, to which P must reply with $f(n,m,P)$. If P replies correctly and on time, then V declares P as being close. The specification rules of DBToy are shown in Figure 7.*

In the *DBToy* rules we have introduced the linear fact symbols Net $\in \Gamma^1$, Send, Recv $\in \Gamma^2$, Action $\in \Gamma^1$ and DBSec $\in \Gamma^4$. A fact $\mathsf{Net}(m)$ denotes that the message m is on the network. A fact $\mathsf{Send}(X,m)$ denotes the sending of m by the agent X, and a fact $\mathsf{Recv}(X,m)$ denotes the reception by X of the message m. A fact $\mathsf{Action}(X)$ denotes that an action was taken by X. A fact $\mathsf{DBSec}(V,P,ch,rp)$ denotes V's claim that P is close during the fast phase, delimited by $\mathsf{Send}(V,ch)$ and $\mathsf{Recv}(V,rp)$. The remaining newly introduced facts denote the agents' information on the system state. Recall that the reserved fact symbols Compromise and Shk are persistent. The rest of the fact symbols used in Figure 7 are linear.

The rules `DBInject`, `DBSend`, and `DBRecv` restrict the Dolev-Yao attackers' communication with protocol participants, in order to comply with the semantic domain of [44]. This was briefly motivated in Section 2. The aim is to capture the statement "every message that can be received by the verifier during the fast phase has been sent from a real physical location". The reason behind this is that messages cannot travel faster than light, thus the adversary cannot instantaneously send a message to an agent (as modelled

$$\mathsf{KeyGen} := \big[\mathsf{Fr}(k)\big]\!\longrightarrow\!\big[\mathsf{Shk}(V,P,k)\big]$$

$$\mathsf{KeyRevV} := \big[\mathsf{Shk}(V,P,k)\big]\xrightarrow{\mathsf{Compromise}(V)}\begin{bmatrix}\mathsf{Out}(k),\\ \mathsf{Compromise}(V)\end{bmatrix}$$

$$\mathsf{KeyRevP} := \big[\mathsf{Shk}(V,P,k)\big]\xrightarrow{\mathsf{Compromise}(P)}\begin{bmatrix}\mathsf{Out}(k),\\ \mathsf{Compromise}(P)\end{bmatrix}$$

$$\mathtt{DBInject} := \big[\mathsf{In}(m),\mathsf{Compromise}(X)\big]\!\longrightarrow\!\big[\mathsf{Send}(X,m)\big]$$

$$\mathtt{DBSend} := \big[\mathsf{Send}(X,m)\big]\xrightarrow{\mathsf{Send}(X,m),\mathsf{Action}(X)}\big[\mathsf{Net}(m),\mathsf{Out}(m)\big]$$

$$\mathtt{DBRecv} := \big[\mathsf{Net}(m)\big]\xrightarrow{\mathsf{Action}(Y),\mathsf{Recv}(Y,m)}\big[\mathsf{Recv}(Y,m)\big]$$

$$\mathsf{P1} := \big[\mathsf{Fr}(m),\ \mathsf{Shk}(V,P,k)\big]\xrightarrow{\mathsf{Start}(m)}\begin{bmatrix}\mathsf{Send}(P,\mathsf{senc}(m,k)),\\ \mathsf{ProvSt1}(P,m)\end{bmatrix}$$

$$\mathsf{V1} := \begin{bmatrix}\mathsf{Fr}(n),\ \mathsf{Shk}(V,P,k),\\ \mathsf{In}(\mathsf{senc}(m,k))\end{bmatrix}\xrightarrow{\mathsf{Start}(n),\mathsf{Send}(V,n)}\begin{bmatrix}\mathsf{Out}(n),\\ \mathsf{VerifSt1}(V,P,n,m)\end{bmatrix}$$

$$\mathsf{P2} := \big[\mathsf{ProvSt1}(P,m),\ \mathsf{In}(n)\big]\xrightarrow{\mathsf{End}(m)}\big[\mathsf{Send}(P,f(n,m,P))\big]$$

$$\mathsf{V2} := \begin{bmatrix}\mathsf{VerifSt1}(V,P,n,m),\\ \mathsf{Recv}(V,f(n,m,P))\end{bmatrix}\xrightarrow{\mathsf{DBSec}(V,P,n,f(n,m,P)),\ \mathsf{End}(n)}\big[\ \big]$$

Figure 7: Specification rules of the *DBToy* protocol.

by Dolev-Yao's rule `Inject` in Figure 3). Hence, in order for the adversary to inject data for an agent to receive, they must use a compromised agent as the sender, and so the message takes a while to arrive to the receiver. Note that we do not drop the traditional `Inject` rule of Figure 3, but we use Recv and Send facts to model the sending and receiving of messages during the fast phase.

In line with this, we will assume that every set of rules defining a distance-bounding protocol is consistent with the usage of Send, Recv and Action facts as follows: (1) every message m sent by the prover P is modelled by a rule with a fact $\mathsf{Send}(P,m)$ within the conclusions, (2) every message m received by the verifier V *during the fast phase* is modelled by a rule with a fact $\mathsf{Recv}(V,m)$ within the premises, and (3) every message m sent by the verifier V *during the fast phase* is modelled by a rule with a fact $\mathsf{Send}(V,m)$ within the actions.

5.2 Secure Distance-Bounding

In [44], Mauw *et al.* defined the causality-based property *secure distance-bounding*, to verify distance-bounding protocols. The property resembles a form of *aliveness* [25, 43] as the prover must perform some action during the fast phase of the protocol. The authors demonstrated that a verifier's guarantee that the prover is alive during the fast phase is equivalent to the verifier's guarantee that the fast phase RTT provides an upper bound to their distance to the prover. Next we formulate Mauw *et al.*'s property:

$$\begin{aligned}
&\mathit{dbsec}(t,l) \iff \\
&\quad \forall V,P,ch,rp.\ \mathsf{DBSec}(V,P,ch,rp) \in t_l \implies \\
&\qquad (\exists i,j,k.\ i<j<k \wedge \mathsf{Send}(V,ch) \in t_i \wedge \\
&\qquad\qquad \mathsf{Action}(P) \in t_j \wedge \mathsf{Recv}(V,rp) \in t_k) \vee \\
&\qquad (\exists b,b',i,j,k,P'. \\
&\qquad\qquad i<j<k \wedge \mathsf{Send}(V,ch) \in t_i \wedge \\
&\qquad\qquad \mathsf{Action}(P') \in t_j \wedge \mathsf{Recv}(V,rp) \in t_k \wedge \\
&\qquad\qquad \mathsf{Compromise}(P) \in t_b \wedge \mathsf{Compromise}(P') \in t_{b'}) \vee \\
&\qquad (\exists i.\ \mathsf{Compromise}(V) \in t_i).
\end{aligned}$$

Secure distance-bounding holds for a trace t if, whenever a claim $\mathsf{DBSec}(V,P,ch,rp)$ occurs, it is the case that there is an action of P (or a compromised prover P' if P is compromised) during the fast phase. Tamarin provides proof of $\mathit{DBToy} \models \mathit{dbsec}$.

Observe that, unlike the agreement property from Section 3, *dbsec* does not exclude traces in which one of the agents involved in the security claim is compromised. Instead, should the prover be compromised, then the verification fails only if no compromised prover is active in the fast phase.

5.3 Formalising (Resistance To) Terrorist Fraud

We informally define terrorist fraud as follows.

Definition 5 (Informal). *Terrorist fraud (TF) is an attack in which a remote and non-compromised prover P colludes with a close and compromised prover A to make the verifier believe that P is close. Conditionally, A (or any other compromised prover) must not be able to attack the protocol again without further collusion.*

The *dbsec* property allows us to detect attacks in which the proving party is compromised, such as distance fraud [28] and distance hijacking [26] (details on these attacks can be found in Appendix A). However, *dbsec* is too fine-grained for modelling terrorist fraud, as we require the distant and colluding prover to be non-compromised (in the case of a compromised prover, collusion actions do little to aid the adversary). In line with this reasoning, we define below a property weaker than *dbsec*, that is conditional on non-compromise of both prover and verifier:

$$\begin{aligned}
&\mathit{dbsec\_hnst}(t,l) \iff \\
&\quad \forall V,P,ch,rp.\ \mathsf{DBSec}(V,P,ch,rp) \in t_l \implies \\
&\qquad (\exists i,j,k.\ i<j<k \wedge \mathsf{Send}(V,ch) \in t_i \wedge \\
&\qquad\qquad \mathsf{Action}(P) \in t_j \wedge \mathsf{Recv}(V,rp) \in t_k) \vee \\
&\qquad (\exists i.\ \mathsf{Compromise}(V) \in t_i \vee \mathsf{Compromise}(P) \in t_i).
\end{aligned}$$

Intuitively, a trace satisfies *dbsec_hnst* if, whenever a verifier V believes a prover P is close, P took some action between the verifier sending the challenge *ch* and receiving reponse *rp*.

We formally define next *resistance to terrorist fraud*, a property formulated by means of post-collusion security with respect to *dbsec_hnst*.

Definition 6 (Resistance to Terrorist Fraud). *A protocol* $\mathit{Proto} \subseteq \mathcal{R} \setminus C$ *is* resistant to terrorist fraud *if every valid extension* Proto' *of Proto that breaks dbsec_hnst is* not *post-collusion secure with respect to dbsec_hnst, i.e.*

$$\mathit{Proto}' \not\models \mathit{dbsec\_hnst} \implies \mathit{Proto}' \not\models^\star \mathit{dbsec\_hnst}. \tag{9}$$

Observe that resistance to terrorist fraud is a property on protocols rather than on traces. Further, terrorist fraud uses the negation of post-collusion security. This is because in a terrorist fraud attack, the colluding prover wishes to allow their partner to complete the protocol only whilst they are cooperating.

5.4 On the Completeness of our Approach

Definition 6 is quantified over all (valid) extensions of a collection of protocol rules. As such, it can present obstacles in providing proofs of security, as the number of extensions is exponential in the complexity of the protocol. Indeed, attempting to fully automate this process is an open problem which is also considered by other approaches [9, 12].

To deal with this completeness issue for the problem of proving terrorist fraud resistance, we introduce the notion of a *least-disclosing* message. Such message is a knowledge-minimal message that the adversary needs, in order to produce the fast phase response upon reception of the challenge. For instance, if *ch* is the verifier's fast phase challenge, and the prover's fast phase response can be written as $f(ch, z_1, \ldots, z_n)$ for some $z_1, \ldots, z_n \in \mathcal{T}_\Sigma(\mathcal{V})$ such that $\lambda ch.f$ is either injective or constant, then a least-disclosing message is $\langle z_1, \ldots, z_n \rangle$. Such message can lead, in some cases, to the disclosure (directly or not) of the long-term keys. To better illustrate the least-disclosing notion, le us consider again the *DBToy* protocol.

Theorem 1. *DBToy is resistant to terrorist fraud.*

Proof. Let DBToy' be a valid extension of *DBToy* such that $\mathit{DBToy}' \not\models \mathit{dbsec\_hnst}$. Thus, there exist $t_1 \cdots t_l \in \mathit{Traces}(\mathit{DBToy}')$, and $n,m,V,P \in \mathcal{T}_\Sigma$, and $i,k \in \{1,\ldots,l\}$ with $i<k$, such that:

$$\begin{aligned}
&\mathsf{Send}(V,n) \in t_i \wedge \mathsf{Recv}(V, f(n,m,P)) \in t_k \wedge \\
&\mathsf{DBSec}(V,P,n,f(n,m,P)) \in t_l \wedge \\
&(\nexists j \in \{i+1,\ldots,k-1\}.\ \mathsf{Action}(P) \in t_j) \wedge \\
&(\nexists j \in \{1,\ldots,l\}.\ \mathsf{Compromise}(V) \in t_j) \wedge \\
&(\nexists j \in \{1,\ldots,l\}.\ \mathsf{Compromise}(P) \in t_j),
\end{aligned} \tag{10}$$

Hence, because of Equation 10 above and given the fact that $\mathsf{Recv}(V, f(n,m,P))$ can only occur due to the rule `DBNet` (see Figure 7), we derive that:

$$\begin{aligned}
&\exists c, j \in \{1,\ldots,k-1\}, C. \\
&\quad (\mathsf{Send}(C, f(n,m,P)) \in t_j \wedge \mathsf{Compromise}(C) \in t_c).
\end{aligned} \tag{11}$$

Equation 11 implies that $\exists w < k.\ \mathsf{K}(m) \in t_w$. This means that DBToy' has a collusion rule in which m is given away. Notice that m (or equivalently $\langle m, P\rangle$) is indeed a *least-disclosing* message because of the following two reasons: m is needed by the adversary to break *dbsec_hnst*, and m is atomic (i.e. it cannot be learned by pieces).

But, if the adversary knows m, then they can use a compromised prover to run again the protocol with V on behalf of P, by using the messages $\mathsf{senc}(m,k)$ and $f(n_2,m,P)$ in that order, where n_2 is V's (new) challenge. This reasoning can be formalized as follows.

Given that $DBToy'$ is valid (see Definition 2) we have that $e \geq l$, and $t_{l+1}, \ldots, t_e$ exist such that:

$$t_1 \cdots t_l \cdots t_e \in Traces(DBToy') \wedge complete(t_1 \cdots t_l \cdots t_e). \quad (12)$$

Now, $l_2 \geq e$, and $t_{e+1}, \ldots, t_{l_2}$, and n_2, and $i_2, k_2 \in \{e+1, \ldots, l_2 - 1\}$ exist such that:

$$\begin{aligned} & t_1 \cdots t_l \cdots t_e \cdots t_{l_2} \in Traces(DBToy') \wedge \\ & \mathsf{Send}(V, n_2) \in t_{i_2} \wedge \mathsf{Recv}(V, f(n_2, m, P)) \in t_{k_2} \wedge \\ & \mathsf{DBSec}(V, P, n_2, f(n_2, m, P)) \in t_{l_2} \wedge \\ & (\nexists j \in \{i_2+1, \ldots, k_2-1\}.\ \mathsf{Action}(P) \in t_j) \wedge \\ & (\nexists j \in \{1, \ldots, l_2\}.\ \mathsf{Compromise}(V) \in t_j) \wedge \\ & (\nexists j \in \{1, \ldots, l_2\}.\ \mathsf{Compromise}(P) \in t_j). \quad (13) \end{aligned}$$

Therefore, from Equations 12 and 13 we deduce that $DBToy' \not\models^{\star} dbsec\_hnst$, which completes the proof[3]. □

The reasoning about the least-disclosing messages is supported by the observation that any follow-up, collusion-free trace which the adversary can lead to with less knowledge, they can also lead to with further knowledge.

6 A SURVEY OF DISTANCE BOUNDING

In this section we show how the Tamarin tool can be used to construct proofs of security or attacks on distance-bounding protocols. In Section 6.1, we describe the results of our verification of a number of distance-bounding protocols. In Section 6.2 we show how the results of our approach differ from those of Chothia *et al.* [20]. Finally, in Section 6.3 we analyse three industrial protocols based on the ISO/IEC 14443 standard and propose Tamarin-verified repaired protocols.

6.1 Case Studies

We conducted verification in Tamarin of a number of distance-bounding protocols from the literature. For each of them, we verify whether it satisfies *dbsec_hnst* (without collusion), whether it satisfies *dbsec* (also without collusion) and whether it resists terrorist fraud (Definition 6). The results are shown in Table 1.

To identify the type of attack against a given protocol, we provide two hints: (1) if the protocol does not satisfy *dbsec_hnst*, then a mafia fraud exists; and (2) if the protocol satisfies *dbsec_hnst* but it does not satisfy *dbsec*, then a distance fraud and/or a distance hijacking exist. In this second case, it is highly recommended to run Tamarin in interactive mode and inspect the trace invalidates the property *dbsec* in order to visually assert the existence of the attack. Further details on this can be found in our repository.

Out of the analysed protocols, only three protocols are distance-bounding secure and resist terrorist fraud. These protocols are Reid *et al.*'s [53], DBPK [17], and Swiss Knife [37]. A total of nineteen protocols were found vulnerable to terrorist fraud.

The authors of UWB impulse radio based protocol [39] do not give precise specifications of their *secure* channel. Hence we employed two schemes: asymmetric encryption/decryption and a message authentication code (MAC). We found a mafia fraud against each variation. Such attack is not reported in [39], as the authors only consider verbatim relay.

For each one of the protocols reported as *not* resistant to terrorist fraud, the valid extension used to invalidate Equation 9 is the prover's leakage of the least-disclosing message, whose notion was discussed in 5.4. For each protocol *Proto* reported as resistant to terrorist fraud, one of the following three cases occurred:

(1) $Proto \not\models dbsec\_hnst$ and $Proto \not\models^{\star} dbsec\_hnst$, thus $Proto' \not\models^{\star} dbsec\_hnst$ for any valid extension $Proto'$ of $Proto$, because $Traces(Proto) \subseteq Traces(Proto')$. The protocols of this type are

[3]A Tamarin proof for a given $DBToy'$ is also available in our repository.

Table 1: Tamarin verification results. We highlighted in bold the protocols that satisfy *dbsec* and resist terrorist fraud. The protocols from the block "Lookup-based" have identical specification. Legend: ✓: verified, ×: attack found, $^{(n)}$: no symbolic, computer-verifiable (in)security proof reported before, $^{(\neq c)}$: differs from Chothia *et al.*'s results [20].

Protocol	Satisfies *dbsec_hnst*	Satisfies *dbsec*	Resists TF
Brands-Chaum [15]			
- Signature id.	✓	×	$\times^{(n)}$
- Fiat-Shamir id.	✓	×	$\times^{(n)}$
CRCS [52]			
- Non-revealing sign.	✓	✓	×
- Revealing sign.	✓	×	×
Meadows *et al.* [46]			
- $f := \langle N_V, P \oplus N_P \rangle$	✓	$\times^{(\neq c)}$	×
- $f := N_V \oplus h(P, N_P)$	$✓^{(n)}$	$✓^{(n)}$	$\times^{(n)}$
- $f := \langle N_V, P, N_P \rangle$	$✓^{(n)}$	$✓^{(n)}$	$\times^{(n)}$
Lookup-based			
- Tree [7]			
- Poulidor [60]	✓	✓	$\times^{(\neq c)}$
- Hancke-Kuhn [34]			
- Uniform [45]			
Munilla-Peinado [48]	✓	✓	$\times^{(n)}$
Kim-Avoine [36]	✓	✓	$\times^{(n)}$
Reid *et al.* [53]	✓	✓	$✓^{(n)}$
MAD (one way) [19]	✓	$\times^{(\neq c)}$	×
DBPK [17]	$✓^{(n)}$	$✓^{(n)}$	$✓^{(n)}$
Swiss Knife [37]	✓	✓	$✓^{(n)}$
UWB [39]			
- Asymmetric	$\times^{(n)}$	$\times^{(n)}$	$✓^{(n)}$
- keyed-MAC	$\times^{(n)}$	$\times^{(n)}$	$✓^{(n)}$
WSBC+DB [50]	$✓^{(n)}$	$\times^{(n)}$	$\times^{(n)}$
Hitomi [51]	$✓^{(n)}$	$✓^{(n)}$	$\times^{(n)}$
TREAD [4]			
- Asymmetric	×	×	$✓^{(n)}$
- Symmetric	✓	×	$✓^{(n)}$
ISO/IEC 14443			
- PaySafe [21]	✓	×	×
- MIFARE Plus [58]	✓	×	×
- PayPass [31]	✓	×	×

TREAD [4] with asymmetric encryption, and both versions of UWB [39].

(2) Every valid extension *Proto′* of *Proto* such that *Proto′* $\not\models$ *dbsec_hnst* leads to replay of an attack on *dbsec_hnst*, therefore *Proto′* $\not\models^{\star}$ *dbsec_hnst*. The protocol of this type is TREAD [4] with symmetric encryption.

(3) Every valid extension *Proto′* of *Proto* such that *Proto′* $\not\models$ *dbsec_hnst* leads to disclosure of the symmetric key shared by the prover and verifier, therefore *Proto′* $\not\models^{\star}$ *dbsec_hnst*. The protocols of this type are Reid *et al.* [53], DBPK [17], and Swiss-Knife [37].

The proofs of terrorist fraud resistance of the four protocols from the last two cases were constructed by following the *semi-automatic* approach based on least-disclosing messages, applied to the *DBToy* protocol proof of Section 5.4. It is a semi-automatic approach, because TAMARIN alone cannot faithfully verify that Definition 6 holds for any protocol. This is because of the complexity of handling the universal quantifier over all valid collusion extensions. However, a simple manual proof analogous to that of the *DBToy* protocol, in combination with the tool successfully led to security proofs of the referred protocols.

On average, a TAMARIN model of a protocol consists of about 260 lines of code, out of which 170 are of generic code, approximately. On a modern laptop, the verification of all lemmas for a given protocol takes about half of a minute on average and a few seconds in most cases. All (in)security proofs were constructed without any proof oracles for speeding up the verification.

6.2 Our Approach vs. Chothia *et al.*'s

As mentioned in Section 2, a recent publication [20] at the *USENIX Security Symposium 2018* analysed a number of distance-bounding protocols. Our findings show incorrectness in their results. In this section we will briefly explain and interpret three inconsistencies between this work and ours. Furthermore, a more detailed discussion can be found in Appendix B.

The first inconsistency is regarding the *DBToy* protocol of Example 2. By employing Chothia *et al.*'s framework, we analysed this protocol[4] and identified a terrorist fraud attack against it. This is in contradiction with Theorem 1. The inconsistency arises from Chothia *et al.* using a different definition of terrorist fraud than us. Their terrorist fraud prover colludes as long as their long-term key(s) are not revealed. It does not consider the (non-)repeatability of the resulting attacks.

Chothia *et al.* [20] also produce results which contradict those widely accepted and reiterated in the literature. For example, Hancke and Kuhn's protocol [34] is widely agreed to admit a terrorist fraud attack (see e.g. [2, 3, 5, 6, 14, 16, 37]). Chothia *et al.* report otherwise, without any discussion on the topic. Such inconsistency is due to an over-approximation on the symbolic abstraction the authors make of the protocol.

Lastly, Chothia *et al.*'s verification results report no attack, other than terrorist fraud, against versions of Meadows *et al.*'s protocol [46] and the MAD protocol [19]. Our verification, however, identifies a valid distance hijacking attack against each of these protocols. We note that such attacks are not new, as they have been reported by a number of previous works, e.g. [6, 26, 27, 44].

[4]Our repository contains the ProVerif model of this and several other protocols.

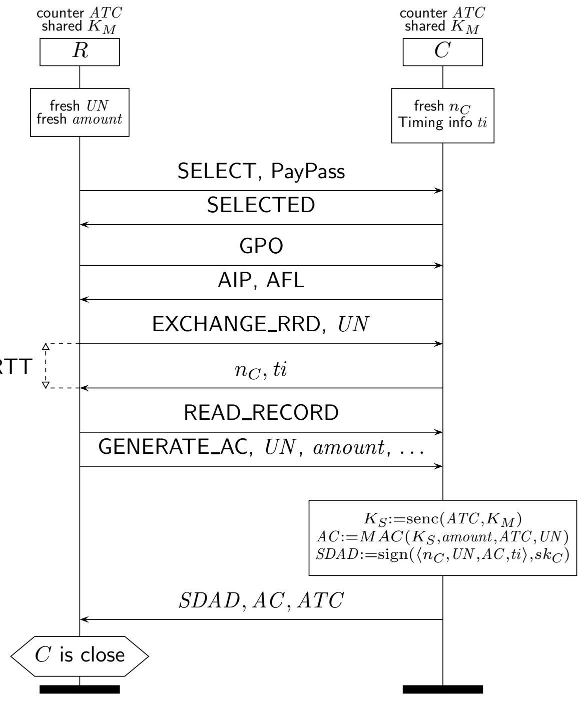


Figure 8: Mastercard's PayPass protocol.

6.3 On the ISO/IEC 14443 Protocols

The ISO/IEC 14443 standard is used in more than 80% of today's contactless smart cards. Within our case studies, we analysed 3 protocols based on this standard. Those protocols are:

- NXP's MIFARE Plus[5] (versions X and EV1) with proximity check (patent [58]) with worldwide applications in public transport, access management, school and campus cards, citizen cards, employee cards, and car parking.
- PaySafe [21], which is a distance-bounding-enabled version of Visa's contactless payment protocol payWave (in qVSDC mode) [32].
- PayPass [31], which is Mastercard's contactless payment protocol with relay resistance.

To demonstrate our analysis, we have chosen the PayPass protocol, represented in Figure 8. The analyses of the other two protocols are analogous. In the context of these protocols, the verifier *R* is the reader terminal and the prover *C* is the card.

PayPass is a relay-resistance-enabled version of the EMV[6] payment protocol implemented in Mastercard's contactless cards. EMV (which stands for Europay, Mastercard and Visa) has become the international standard for smart cards/chips payment protocols.

In a regular EMV session, a transaction is initiated by the exchange of SELECT and SELECTED commands along with the selected EMV applet that will be used for the transaction (PayPass in

[5]https://www.mifare.net/en/products/chip-card-ics/mifare-plus
[6]https://www.emvco.com

this case). Then, the terminal issues the GPO command to inform the card on the terminal's capabilities. The card then responds to this command with the Application Interchange Profile (AIP) and Application File Locator (AFL) which indicate the card's capabilities and the location of data files, respectively. Then, the terminal issues the GENERATE_AC command, which includes an Unpredictable Number *UN*, the amount of the transaction, the currency code, and other data. The cards responds with the Application Cryptogram (*AC*), the Signed Dynamic Application Data (*SDAD*) and the Application Transaction Counter (*ATC*). The *AC* is a the result of keyed-MAC on the transaction information whose key is an encryption of the Application Transaction Counter (*ATC*, equal to the number of transactions previously made by the card) with a long-term symmetric key shared between the terminal and the card. The *AC* is the proof of the transaction, which can be verified by the card issuer. The *SDAD* is the card's signature of the transaction information.

To ensure the EMV protocol satisfies relay resistance, after the AIP and AFL commands, the terminal issues the new Exchange Relay Resistance Data EXCHANGE_RRD command, along with the Terminal Relay Resistance Entropy number (which equals *UN*). This message initiates the fast phase of the protocol. The card must respond on time with their nonce n_C (Device Relay Resistance Entropy) and three timing estimates: minimum time for processing, maximum time for processing and estimated transmission time.

When modelling the PayPass protocol in Tamarin, and also the other ISO/IEC 14443 protocols, we made the following abstractions: (1) the timing information is considered a nonce; and (2) we did not model any exchanged messages that are fully composed of constant terms, e.g. the first message $\langle \mathsf{SELECT}, \mathrm{PayPass}\rangle$.

As Table 1 shows, PayPass satisfies *dbsec_hnst*, which means that it does resist mafia fraud and in particular, relay attacks. Indeed, defending against relay is a fundamental security goal of this protocol. However, PayPass fails to defend against distance fraud [28] and distance hijacking [26]. Those attacks refer to a remote and compromised card which successfully tricks the reader into believing they are co-located, and thus the reader accepts the transaction.

One might argue that those attacks are irrelevant for payment systems. After all, it is the compromised card's owner's bank account which ends up being charged. However, suppose an attacker has acquired the payment card of a victim and wishes to cause them harm. After compromising the card, they might place a concealed device near the checkout area of a store that performs a distance hijacking attack using the compromised card. Shoppers at the store would then perform transactions, believing that they were paying for products, whilst in fact all payments came from the one corrupted card. The attacker could even mix in several transactions of their own, which would be indistinguishable from the honest shoppers. As a result of this "Robin Hood" style attack, the victim will be charged for these illegitimate transactions with no clear perpetrator.

Fixing the ISO/IEC 14443 Protocols. As before, we will focus on the PayPass protocol. Before giving the fixes, let us motivate the reasons for which it does not satisfy *dbsec*. As noted by Mauw *et al.* in [44] when analysing PaySafe, a distance fraud is possible due to the lack of a causal relation between the fast phase challenge and response. That is, the fast phase response can be produced prior to reception of the challenge. To solve this issue, Mauw *et al.* suggested the inclusion of the reader's nonce *UN* within the card's response.

Mauw *et al.*'s suggestion applied to PayPass does prevent distance fraud, but it does not prevent distance hijacking. To prevent the latter, we must bind the fast phase messages to the card's identity. We do so by adding to the card's fast phase response, besides *UN*, the card's signature on the nonce n_C. Thus, the card's fast phase response becomes:

$$\langle n_C, ti, \mathrm{sign}(n_C, sk_C), UN\rangle\,.$$

This modification results in a protocol PayPass_Fix that satisfies *dbsec*. Observe that the signature $\mathrm{sign}(n_C, sk_C)$ can be computed prior to the fast phase, so it does not delay the card's response.

The very same solution of adding $\langle \mathrm{sign}(n_C, sk_C), UN\rangle$ into the card's fast phase response works for both the PaySafe and MIFARE Plus protocols as well. Though, to keep consistency with the usage of cryptographic operations in the case of the latter protocol, we propose a keyed-MAC message $MAC(K_M, n_C, \text{`1'}, \text{`2'})$ instead of the signature $\mathrm{sign}(n_C, sk_C)$. As before, the keyed-MAC message can be computed prior to the fast phase.

The modified protocol PayPass_Fix does not resist terrorist fraud because the card's leakage of $\langle n_C, ti, \mathrm{sign}(n_C, sk_C)\rangle$ prior to the fast phase leads to a valid attack. To prevent terrorist fraud, we propose to further modify the PayPass protocol by changing the card's fast phase response and *SDAD* messages so that they become:

$$\langle n_C, ti, f(UN, n_C \oplus K_M)\rangle \quad \text{and} \quad \mathrm{sign}(\langle UN, AC\rangle, sk_C),$$

respectively; where f is an irreversible function. The referred modification on PayPass results in a protocol PayPass_FixTF that satisfies *dbsec* and resists terrorist fraud.

Similar constructions can be performed on PaySafe and MIFARE Plus in order to repair them. The Tamarin models and security proofs of the two versions of each protocol are available in our repository. We give two different repaired versions of each protocol in order to leave the choice up to the requirements of the application system. For example, if terrorist fraud is not a critical issue, then the first modification (i.e. Protocol_Fix) is suggested over the second one (i.e. Protocol_FixTF) as the latter modifies the standard more "aggressively". We do always suggest the first modified version over the original protocol, regardless of the application.

Other modifications for the ISO/IEC 14443 protocols that make them resistant to terrorist fraud possibly exist, and likely all of them (like ours) would require major changes to the standard. For example, the composition of the *SDAD* message would likely have to be modified due to the occurrence of the card's nonces within it. Furthermore, we conjecture that if the card's nonces (e.g. n_C) can be inferred from passive observation of the execution, then versions of the protocols in question that resist terrorist fraud would be vulnerable to relay attacks, thus violating a primary security goal of these protocols.

7 CONCLUSIONS

This paper addressed symbolic analysis of security protocols in the presence of colluding agents. Colluding agents are agents who are not under full control of the adversary, yet they are willing to deviate from the intended protocol execution with the goal to invalidate a given property. By looking at different use-cases, we observe that post-collusion security may or may not be a desirable goal. This is because the risk of irreparable damage to the security of a protocol may motivate agents to avoid collusion.

We proposed a concrete symbolic formulation of post-collusion security that can be implemented in state-of-the-art protocol verification tools such as Tamarin. We used our definition to illustrate that leakage of session data can lead to impersonation of agents. This is particularly interesting in the context of authentication properties in which agents, by leaking only session-fresh data, enable the adversary to successfully break the authentication property in every session thereafter.

By means of post-collusion security, we provided the first formal symbolic definition of (resistance to) the sophisticated terrorist fraud attack against distance-bounding protocols. By using our theoretical model and the Tamarin tool, we provided computer-verifiable proofs of the (in)security of over 25 distance-bounding protocols that account for all classes of attacks, as given by the literature on distance bounding. To the best of our knowledge, this is the most extensive and sound set of security/vulnerability proofs within this research subject.

Our verification reports that for the vast majority of the analysed protocols at least one attack exists. The vulnerable protocols include protocols based on the ISO/IEC 14443 standard such as Mastercard's PayPass [31], Visa's payWave with distance-bounding [21], and NXP's MIFARE Plus with proximity check [58]. Finally, we proposed fixes for these protocols and provide computer-verifiable security proofs of the repaired protocols. The proposed fixes form demonstrative examples that could be used to improve proximity-based secure systems that follow the standard, or may even form guidance for a future version of the standard itself.

A ATTACKS ON DISTANCE BOUNDING

There exist four main classes of attacks on distance-bounding protocols. Some authors consider more classes but, consistent with [26], our classification represents an exhaustive partition of the full space of attacks: *mafia fraud* [29], *distance fraud* [28], *distance hijacking* [26], and *terrorist fraud* [29]. We briefly describe next these attacks, and their graphical representations are shown in Figure 9.

Mafia Fraud (Figure 9a) Given a verifier V, a close compromised agent A uses a distant and honest prover P to make V believe that P is close. The attack works in two sessions: one between A and P (the so-called *pre-ask* session) and another one between V and A. In most cases, A relays the verbatim untimed communication between P and V, and impersonates P to V during the fast phase.

Distance Fraud (Figure 9b) Given a verifier V, a distant and compromised prover P anticipates V's challenges so that V believes that P is close. This means that P must be able to reproduce the responses prior to receiving the challenges. For this type of attack, P does not need to use any other provers.

Distance Hijacking (Figure 9c) Given a verifier V, a distant and compromised prover P makes use of a close and honest prover A to make V believe that P is close. To achieve this, P lets A run the fast phase with V. Then (or, in some protocols, before) P hijacks the session, injecting their own identity-defining messages, possibly during the verification phase of the protocol. This makes V believe that they are running the protocol, and in particular the fast phase, with P.

Terrorist Fraud (Figure 9d) Given a verifier V, a close and compromised prover A, and a distant (non-compromised) prover P collude to make V believe that P is close. A condition for this attack to be valid is that, without further collusion involving A, it must not be possible to further convince V that A is close.

B CHOTHIA *ET AL.*'S FRAMEWORK VS. OURS

In this appendix we provide a detailed description on the inconsistencies between Chothia *et al.*'s work [20] in relation to our framework and results.

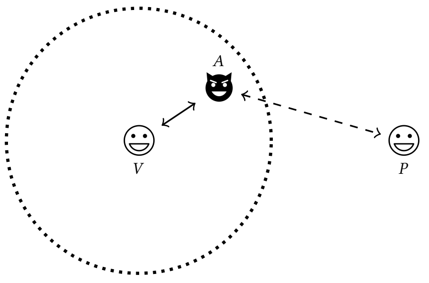
B.1 On Terrorist Fraud

Chothia *et al.*'s terrorist fraud approach and ours yield different results when analysing the *DBToy* protocol in Example 2. The reason is a fundamental difference between the approaches in defining what consists a terrorist fraud attack.

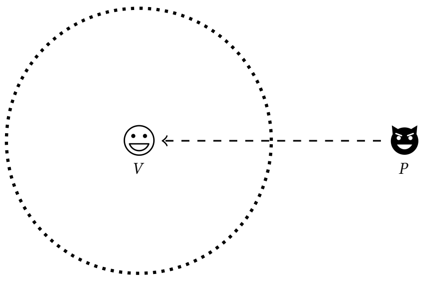
Both definitions state that a distant prover, by colluding with a close and compromised prover, make a verifier believe that the distant prover is close. The definitions differ in the condition on the collusion that make one consider the attack as valid. Chothia *et al.*'s definition states that a terrorist fraud occurs whenever the distant prover does not reveal their secret keys in the process of collusion. Our definition, on the other hand, requires a stronger condition, in that the distant prover must not allow the compromised prover to proof proximity in further sessions without further collusion.

The reason for this is related to the reveal of messages that are as relevant as secret keys, rather than the reveal of secret keys themselves. For example, for some protocols, a prover's leakage of session-fresh data can lead to their impersonation in every session thereafter. A running example is given as follows. Consider a distance-bounding protocol *Proto* in which every crypto-operation uses a shared, symmetric key. Let us assume that the only aid the distant prover can provide the close prover with is to give away the shared key. If we follow Chothia *et al.*'s approach, *Proto* would be resistant to terrorist fraud. Consider now another protocol *Proto′* that results from *Proto* by replacing any instance of a shared key k by its hash $h(k)$. Therefore, if we use Chothia *et al.*'s approach, *Proto′* would *not* be resistant to terrorist fraud, as the distant prover can leak $h(k)$, which does not reveal k itself. In this case, the message $h(k)$ is equally as valuable as the key k itself. This means that the key-hashing transformation weakens the protocol, which does not seem to be a coherent statement. The mentioned issue does not occur if we apply our approach as both *Proto* and *Proto′* would be resistant to terrorist fraud.

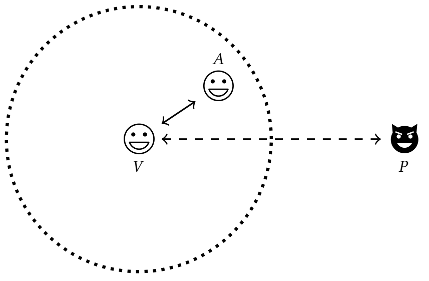
Another inconsistency between our results on terrorist fraud analysis and those of Chothia *et al.* is with respect to Hancke and Kuhn's protocol [34], depicted in Figure 10. This protocol was reported as resistant to terrorist fraud by Chothia *et al.*'s ProVerif



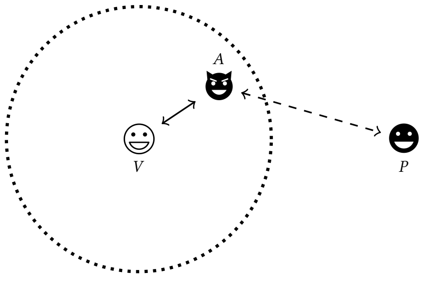
(a) Mafia fraud



(b) Distance fraud



(c) Distance hijacking



(d) Terrorist fraud

Figure 9: Types of attack on distance-bounding protocols. In all cases, V is the verifier, P is the prover who V claims proximity to, and A is an agent in the proximity of V. Filled icons represent agents who either collude or are compromised by the adversary. Unfilled icons represent honest agents, i.e. agents who don't deviate from the protocol specification. The encircled area represents V's proximity, which is bounded by the predefined threshold on the round-trip time. Dashed arrows represent untimed communication, which is communication that does not occur entirely within the fast phase. Thanks to https://thenounproject.com for the icons.

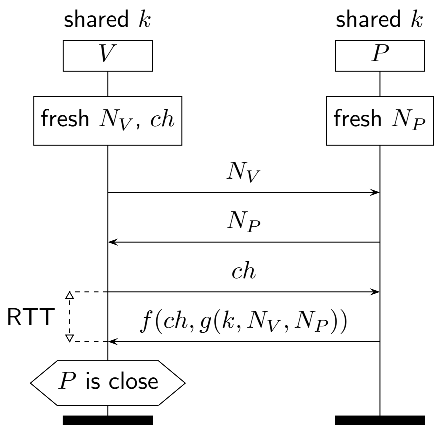


Figure 10: Hancke and Kuhn's protocol [34]. In Chothia *et al.* [20] the prover's response $f(ch, g(k, N_V, N_P))$ is modelled as $f(ch, N_V, N_P, k)$. This over-approximated modelling is the cause for which Chothia *et al.*'s verification results did not identify a terrorist fraud attack against this protocol.

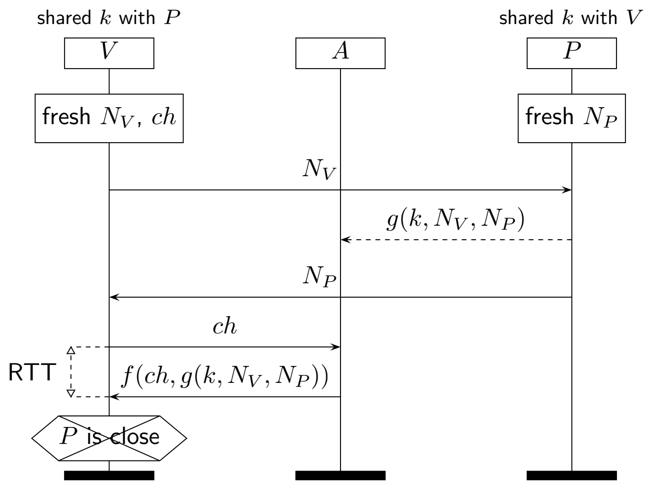


Figure 11: A terrorist fraud attack on Hancke and Kuhn's protocol [34]. The distant prover P colludes with the close and compromised prover A by giving away $g(k, N_V, N_P)$, represented by the dashed arrow. This allows A to impersonate P to the verifier V during the fast phase, hence V believes that P is close. The same false-proximity proof cannot be issued in further sessions without further collusion because $g(k, N_V, N_P)$ is fresh in every session and k is not revealed.

verification [20]. We refute Chothia *et al.*'s statement by demonstrating a trivial terrorist attack, represented in Figure 11.

Recall that in Chothia *et al.* [20], the prover's response message is modelled as $f(ch, N_V, N_P, k)$ which is an inaccurate modelling

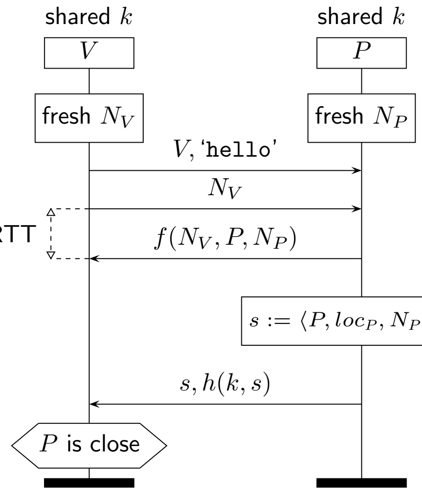


Figure 12: Meadows *et al.*'s protocol [46], where loc_P denotes the location of the prover P. Such location has no impact on any symbolic analysis as it comes in plain text, thus it is modelled as a nonce. Three instances of the protocol are proposed by the authors, given by the following three choices of f: $\langle N_V, P \oplus N_P \rangle$, $\langle N_V, P, N_P \rangle$, and $N_V \oplus h(P, N_P)$ where $\oplus$ is the exclusive-OR and h is a collision-free hash function.

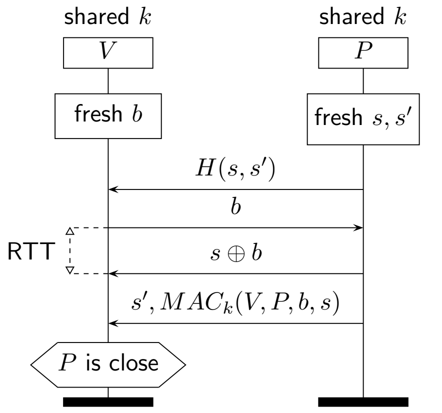


Figure 13: Capkun *et al.*'s MAD protocol [19]. This is the variant with prover-to-verifier authentication. H is a hash function and MAC_k is a keyed-MAC function.

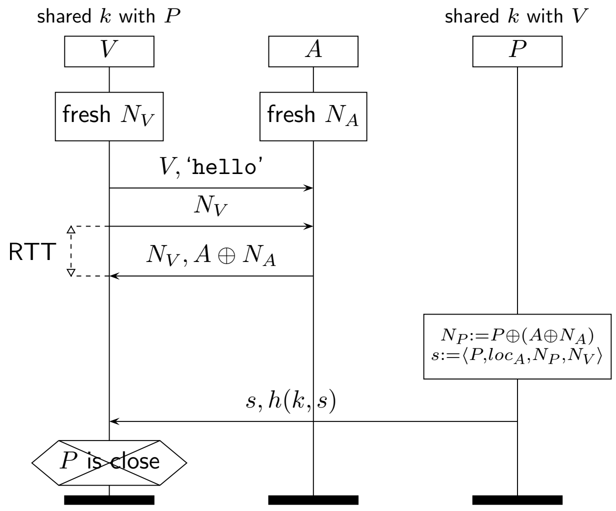


Figure 14: A distance hijacking attack on Meadows *et al.*'s protocol with the choice $f := \langle N_V, P \oplus N_P \rangle$. P learns N_V and $A \oplus N_A$ from passive observation of the legitimate fast phase. Hence, P produces their own authentication message $\langle s, h(k, s) \rangle$, given that the equality $P \oplus N_P = A \oplus N_A$.

compared to the original protocol specification. We remark that the order of the arguments is not the issue, but the *level* at which they occur in the function f. In particular, the prover uses the values N_V, N_P and k to seed a PRNG g before the fast phase begins. During the phase phase, the prover then uses this PRNG to respond to challenges. As a result, a more faithful representation of the challenge response message is $f(ch, g(k, N_V, N_P))$, in which the shared key k is not in the same level as the challenge ch. As a result, the prover can leak $g(k, N_V, N_P)$ (i.e. the seeded PRNG) without leaking k. In the representation of Chothia *et al.*, k and ch are at the same level, hence in order for the prover to leak *dbsec_hnst*-breaking data, the shared key k must be leaked, and this indeed makes the attack no longer valid. This is the reason for which Chothia *et al.* do not deliver the attack depicted Figure 11.

B.2 On Distance Hijacking

Chothia *et al.*'s verification reports no attack, other than terrorist fraud, against Meadows *et al.*'s protocol [46] version in which the prover's fast phase response is $\langle N_V, P \oplus N_P \rangle$, and the MAD protocol [19] with one-way authentication. These protocols are depicted in Figures 12 and 13, respectively.

Our verification, however, identifies a valid distance hijacking attack against each of these protocols (see Figures 14 and 15). In both cases, the compromised prover P, who is distant from the verifier V, hijacks the session between V and a close-by and honest prover A by replacing the final authentication message of the legitimate and close P with their own authentication message, thus making V believe that P is close. In the case of Meadows *et al.*'s protocol, the existence of this attack is indeed consistent with the authors' own statement that their model does not cover attacks of the compromised-prover type. As previously mentioned in Section 6.2 before, a number of previous papers, such as [6, 26, 27, 44], also report distance hijacking attacks against these two protocols.

B.3 Attempts to Implement Distance Hijacking

The checkDB[7] tool created by Chothia *et al.* [20] is used to compile ProVerif files for distance bounding protocols. We attempted to find distance hijacking attacks by using the checkDB tool as follows. A class of attacks is defined in Chothia *et al.*'s model by a configuration of agents. For example, the configuration $[V(id)] \| [DP(id)]$ corresponds to a verifier V authenticating a distant dishonest prover

[7] http://www.cs.bham.ac.uk/~tpc/distance-bounding-protocols/CheckDB-master.tar.gz

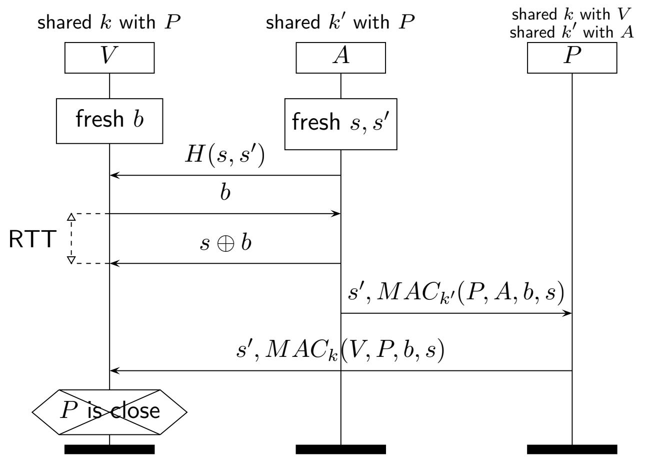


Figure 15: A distance hijacking attack on MAD protocol [19] with one-way authentication. P learns b and $s \oplus b$ (and s by extension) from passive observation of the legitimate fast phase. Therefore, P produces their own authentication message $\langle s', MAC_k(V, P, b, s)\rangle$.

DP - this represents a distance fraud attack. In the syntax of the checkDB tool, this is written as:

```
[!Verifier] |
[!(new id;
  let idP = id in !DishonestProver)
]
```

Security is then modelled by the reachability of the query:

```
query ev:Verified(id)
```

Intuitively, the query is satisfied whenever it is possible for a verifier to reach the end of their role specification (the value *id* is a free term and not tied to a specific identifier). Note that in the case of distance fraud, this means the query is satisfied only if there is an attack.

The checkDB tool is also used to verify terrorist fraud, mafia fraud, and a property referred to as *uncompromised distance bounding*. Note that in each of these cases, the configuration also ensures that a verifier can only reach the end of their specification if an attack exists.

We attempted to implement distance hijacking by testing the process:

```
[ !Verifier |
  (new id2;
  let idP=id2 in !Prover)
] |
[ !(new id;
  let idP=id in !DishonestProver)
]
```

However, this naive implementation leads to several problems when applying it to various protocols:

- In this configuration, 'honest' sessions can be completed between the verifier and local prover. The non-specificity of the query can lead to these honest traces being flagged as potential attacks.
- Many protocols which admit distance hijacking attacks involve the prover revealing their identity only late in the protocol, by use of a shared key or signing key. One workaround - using a dummy message to indicate to a verifier who their partner is - affects the faithfulness of the model and can lead to the false attacks mentioned above.
- Several distance hijacking attacks (including Meadows and MAD) rely on the adversary abusing the algebraic properties of the exclusive-or operator. It is not clear if the equational theory used in the given models (which relies on applying deconstructor functions) is sufficient to model a prover being unable to distinguish e.g. $x \oplus y$ and $x \oplus z \oplus z \oplus y$.

We emphasize that while the *model* of Chothia *et al.* distinguishes which prover in a configuration is the one being 'tested' for proximity, their *tooling* does not make this distinction without manually modifying compiled output files. This difference between model and implementation has no impact in configurations in which honest executions cannot complete, but this is not the case for distance hijacking attacks.

ACKNOWLEDGMENTS

This work was partially supported by the Luxembourg National Research Fund (FNR) under the grants AFR-PhD-10188265 and C15-IS-10428112.

REFERENCES

[1] Yonatan Aumann and Yehuda Lindell. 2010. Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. *J. Cryptology* 23, 2 (2010), 281–343. https://doi.org/10.1007/s00145-009-9040-7

[2] Gildas Avoine, Muhammed Ali Bingöl, Ioana Boureanu, Srdjan Capkun, Gerhard P. Hancke, Süleyman Kardas, Chong Hee Kim, Cédric Lauradoux, Benjamin Martin, Jorge Munilla, Alberto Peinado, Kasper Bonne Rasmussen, Dave Singelée, Aslan Tchamkerten, Rolando Trujillo-Rasua, and Serge Vaudenay. 2019. Security of Distance-Bounding: A Survey. *ACM Comput. Surv.* 51, 5 (2019), 94:1–94:33. https://dl.acm.org/citation.cfm?id=3264628

[3] Gildas Avoine, Muhammed Ali Bingöl, Süleyman Kardas, Cédric Lauradoux, and Benjamin Martin. 2011. A framework for analyzing RFID distance bounding protocols. *Journal of Computer Security* 19, 2 (2011), 289–317. https://doi.org/10.3233/JCS-2010-0408

[4] Gildas Avoine, Xavier Bultel, Sébastien Gambs, David Gérault, Pascal Lafourcade, Cristina Onete, and Jean-Marc Robert. 2017. A Terrorist-fraud Resistant and Extractor-free Anonymous Distance-bounding Protocol. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017.* 800–814. https://doi.org/10.1145/3052973.3053000

[5] Gildas Avoine, Cédric Lauradoux, and Benjamin Martin. 2011. How secret-sharing can defeat terrorist fraud. In *Proceedings of the Fourth ACM Conference on Wireless Network Security, WISEC 2011, Hamburg, Germany, June 14-17, 2011.* 145–156. https://doi.org/10.1145/1998412.1998437

[6] Gildas Avoine, Sjouke Mauw, and Rolando Trujillo-Rasua. 2015. Comparing distance bounding protocols: A critical mission supported by decision theory. *Computer Communications* 67 (2015), 92–102. https://doi.org/10.1016/j.comcom.2015.06.007

[7] Gildas Avoine and Aslan Tchamkerten. 2009. An Efficient Distance Bounding RFID Authentication Protocol: Balancing False-Acceptance Rate and Memory Requirement. In *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings.* 250–261. https://doi.org/10.1007/978-3-642-04474-8_21

[8] David A. Basin, Srdjan Capkun, Patrick Schaller, and Benedikt Schmidt. 2009. Let's Get Physical: Models and Methods for Real-World Security Protocols. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings.* 1–22. https://doi.org/10.1007/978-3-642-03359-9_1

[9] David A. Basin and Cas Cremers. 2014. Know Your Enemy: Compromising Adversaries in Protocol Analysis. *ACM Trans. Inf. Syst. Secur.* 17, 2 (2014), 7:1–7:31. https://doi.org/10.1145/2658996

[10] David A. Basin and Cas J. F. Cremers. 2010. Modeling and Analyzing Security in the Presence of Compromising Adversaries. In *Computer Security - ESORICS 2010, 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings.* 340–356. https://doi.org/10.1007/978-3-642-15497-3_21

[11] David A. Basin, Jannik Dreier, Lucca Hirschi, Sasa Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018.* 1383–1396. https://doi.org/10.1145/3243734.3243846

[12] David A. Basin, Sasa Radomirovic, and Lara Schmid. 2016. Modeling Human Errors in Security Protocols. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016.* 325–340. https://doi.org/10.1109/CSF.2016.30

[13] Thomas Beth and Yvo Desmedt. 1990. Identification Tokens - or: Solving the Chess Grandmaster Problem. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings.* 169–177. https://doi.org/10.1007/3-540-38424-3_12

[14] Ioana Boureanu, Aikaterini Mitrokotsa, and Serge Vaudenay. 2013. Towards Secure Distance Bounding. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers.* 55–67. https://doi.org/10.1007/978-3-662-43933-3_4

[15] Stefan Brands and David Chaum. 1993. Distance-Bounding Protocols (Extended Abstract). In *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings.* 344–359. https://doi.org/10.1007/3-540-48285-7_30

[16] Agnès Brelurut, David Gerault, and Pascal Lafourcade. 2015. Survey of Distance Bounding Protocols and Threats. In *Foundations and Practice of Security - 8th International Symposium, FPS 2015, Clermont-Ferrand, France, October 26-28, 2015, Revised Selected Papers.* 29–49. https://doi.org/10.1007/978-3-319-30303-1_3

[17] Laurent Bussard and Walid Bagga. 2005. Distance-Bounding Proof of Knowledge to Avoid Real-Time Attacks. In *Security and Privacy in the Age of Ubiquitous Computing, IFIP TC11 20th International Conference on Information Security (SEC 2005), May 30 - June 1, 2005, Chiba, Japan.* 223–238.

[18] Ran Canetti and Rafail Ostrovsky. 1999. Secure Computation with Honest-Looking Parties: What If Nobody Is Truly Honest? (Extended Abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA.* 255–264. https://doi.org/10.1145/301250.301313

[19] Srdjan Capkun, Levente Buttyán, and Jean-Pierre Hubaux. 2003. 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 2003, Fairfax, Virginia, USA, 2003.* 21–32. https://doi.org/10.1145/986858.986862

[20] Tom Chothia, Joeri de Ruiter, and Ben Smyth. 2018. Modelling and Analysis of a Hierarchy of Distance Bounding Attacks. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.* 1563–1580. https://www.usenix.org/conference/usenixsecurity18/presentation/chothia

[21] Tom Chothia, Flavio D. Garcia, Joeri de Ruiter, Jordi van den Breekel, and Matthew Thompson. 2015. Relay Cost Bounding for Contactless EMV Payments. In *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers.* 189–206. https://doi.org/10.1007/978-3-662-47854-7_11

[22] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2018. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018.* 1802–1819. https://doi.org/10.1145/3243734.3243747

[23] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. 2016. On Post-compromise Security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016.* 164–178. https://doi.org/10.1109/CSF.2016.19

[24] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. 2017. A Comprehensive Symbolic Analysis of TLS 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017.* 1773–1788. https://doi.org/10.1145/3133956.3134063

[25] Cas Cremers and Sjouke Mauw. 2012. *Operational Semantics and Verification of Security Protocols.* Springer. https://doi.org/10.1007/978-3-540-78636-8

[26] Cas J. F. Cremers, Kasper Bonne Rasmussen, Benedikt Schmidt, and Srdjan Capkun. 2012. Distance Hijacking Attacks on Distance Bounding Protocols. In *IEEE Symposium on Security and Privacy, S&P 2012, 21-23 May 2012, San Francisco, California, USA.* 113–127. https://doi.org/10.1109/SP.2012.17

[27] Alexandre Debant, Stéphanie Delaune, and Cyrille Wiedling. 2018. A Symbolic Framework to Analyse Physical Proximity in Security Protocols. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India.* 29:1–29:20. https://doi.org/10.4230/LIPIcs.FSTTCS.2018.29

[28] Yvo Desmedt. 1988. Major Security Problems with the "Unforgeable" (Feige)-Fiat-Shamir Proofs of Identity and How to Overcome them. In *SECURICOM'88.* 15–17.

[29] Yvo Desmedt, Claude Goutier, and Samy Bengio. 1987. Special Uses and Abuses of the Fiat-Shamir Passport Protocol. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings.* 21–39. https://doi.org/10.1007/3-540-48184-2_3

[30] Danny Dolev and Andrew Chi-Chih Yao. 1983. On the security of public key protocols. *IEEE Trans. Information Theory* 29, 2 (1983), 198–207. https://doi.org/10.1109/TIT.1983.1056650

[31] EMVCo. 2018. *EMV Contactless Specifications for Payment Systems, Book C-2, Kernel 2 Specification, Version 2.7.*

[32] EMVCo. 2018. *EMV Contactless Specifications for Payment Systems, Book C-3, Kernel 3 Specification, Version 2.7.*

[33] Matthew K. Franklin and Moti Yung. 1992. Communication Complexity of Secure Computation (Extended Abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada.* 699–710. https://doi.org/10.1145/129712.129780

[34] Gerhard P. Hancke and Markus G. Kuhn. 2005. An RFID Distance Bounding Protocol. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks, SecureComm 2005, Athens, Greece, 5-9 September, 2005.* 67–73. https://doi.org/10.1109/SECURECOMM.2005.56

[35] Martin Hirt and Ueli M. Maurer. 1997. Complete Characterization of Adversaries Tolerable in Secure Multi-Party Computation (Extended Abstract). In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997.* 25–34. https://doi.org/10.1145/259380.259412

[36] Chong Hee Kim and Gildas Avoine. 2009. RFID Distance Bounding Protocol with Mixed Challenges to Prevent Relay Attacks. In *Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings.* 119–133. https://doi.org/10.1007/978-3-642-10433-6_9

[37] Chong Hee Kim, Gildas Avoine, François Koeune, François-Xavier Standaert, and Olivier Pereira. 2008. The Swiss-Knife RFID Distance Bounding Protocol. In *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers.* 98–115. https://doi.org/10.1007/978-3-642-00730-9_7

[38] Hugo Krawczyk. 2005. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings.* 546–566. https://doi.org/10.1007/11535218_33

[39] Marc Kuhn, Heinrich Luecken, and Nils Ole Tippenhauer. 2010. UWB impulse radio based distance bounding. In *7th Workshop on Positioning Navigation and Communication, WPNC 2010, Dresden Germany, 11-12 March 2010, Proceedings.* 28–37. https://doi.org/10.1109/WPNC.2010.5653801

[40] Brian A. LaMacchia, Kristin E. Lauter, and Anton Mityagin. 2007. Stronger Security of Authenticated Key Exchange. In *Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings.* 1–16. https://doi.org/10.1007/978-3-540-75670-5_1

[41] Kristin E. Lauter and Anton Mityagin. 2006. Security Analysis of KEA Authenticated Key Exchange Protocol. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings.* 378–394. https://doi.org/10.1007/11745853_25

[42] Gavin Lowe. 1995. An Attack on the Needham-Schroeder Public-Key Authentication Protocol. *Inf. Process. Lett.* 56, 3 (1995), 131–133. https://doi.org/10.1016/0020-0190(95)00144-2

[43] Gavin Lowe. 1997. A Hierarchy of Authentication Specifications. In *10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA.* 31–44. https://doi.org/10.1109/CSFW.1997.596782

[44] Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. 2018. Distance-Bounding Protocols: Verification without Time and Location. In *2018 IEEE Symposium on Security and Privacy, S&P 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA.* 549–566. https://doi.org/10.1109/SP.2018.00001

[45] Sjouke Mauw, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. 2016. A Class of Precomputation-Based Distance-Bounding Protocols. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016.* 97–111. https://doi.org/10.1109/EuroSP.2016.19

[46] Catherine A. Meadows, Radha Poovendran, Dusko Pavlovic, LiWu Chang, and Paul F. Syverson. 2007. Distance Bounding Protocols: Authentication Logic Analysis and Collusion Attacks. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks.* 279–298. https://doi.org/10.1007/978-0-387-46276-9_12

[47] Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. 2013. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings.* 696–701. https://doi.org/10.1007/978-3-642-39799-8_48

[48] Jorge Munilla and Alberto Peinado. 2008. Distance bounding protocols for RFID enhanced by using void-challenges and analysis in noisy channels. *Wireless Communications and Mobile Computing* 8, 9 (2008), 1227–1232. https://doi.org/

10.1002/wcm.590

[49] Roger M. Needham and Michael D. Schroeder. 1978. Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM* 21, 12 (1978), 993–999. https://doi.org/10.1145/359657.359659

[50] Pedro Peris-Lopez, Julio César Hernández-Castro, Juan M. Estévez-Tapiador, Esther Palomar, and Jan C. A. van der Lubbe. 2010. Cryptographic puzzles and distance-bounding protocols: Practical tools for RFID security. In *2010 IEEE International Conference on RFID (IEEE RFID 2010)*. 45–52. https://doi.org/10.1109/RFID.2010.5467258

[51] Pedro Peris-Lopez, Julio César Hernández-Castro, Juan M. Estévez-Tapiador, and Jan C. A. van der Lubbe. 2009. Shedding Some Light on RFID Distance Bounding Protocols and Terrorist Attacks. *CoRR* abs/0906.4618 (2009). arXiv:0906.4618 http://arxiv.org/abs/0906.4618

[52] Kasper Bonne Rasmussen and Srdjan Capkun. 2010. Realization of RF Distance Bounding. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*. 389–402. http://www.usenix.org/events/sec10/tech/full_papers/Rasmussen.pdf

[53] Jason Reid, Juan Manuel González Nieto, Tee Tang, and Bouchra Senadji. 2007. Detecting relay attacks with timing-based protocols. In *Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2007, Singapore, March 20-22, 2007*. 204–213. https://doi.org/10.1145/1229285.1229314

[54] Patrick Schaller, Benedikt Schmidt, David A. Basin, and Srdjan Capkun. 2009. Modeling and Verifying Physical Properties of Security Protocols for Wireless Networks. In *Proceedings of the 22nd IEEE Computer Security Foundations Symposium, CSF 2009, Port Jefferson, New York, USA, July 8-10, 2009*. 109–123. https://doi.org/10.1109/CSF.2009.6

[55] Benedikt Schmidt, Simon Meier, Cas J. F. Cremers, and David A. Basin. 2012. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*. 78–94. https://doi.org/10.1109/CSF.2012.25

[56] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (1979), 612–613. https://doi.org/10.1145/359168.359176

[57] Paul Syverson, Catherine Meadows, and Iliano Cervesato. 2000. Dolev-Yao is no better than Machiavelli. In *First Workshop on Issues in the Theory of Security, WITS'00, Geneva, Switzerland, July 7-8, 2000*. 87–92. https://www.nrl.navy.mil/itd/chacs/syverson-dolev-yao-no-better-machiavelli

[58] Peter Thueringer, Hans De Jong, Bruce Murray, Heike Neumann, Paul Hubmer, and Susanne Stern. 2011. Decoupling of measuring the response time of a transponder and its authentication. US Patent No. US12994541.

[59] Martin Tompa and Heather Woll. 1986. How to Share a Secret with Cheaters. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*. 261–265. https://doi.org/10.1007/3-540-47721-7_20

[60] Rolando Trujillo-Rasua, Benjamin Martin, and Gildas Avoine. 2010. The Poulidor Distance-Bounding Protocol. In *Radio Frequency Identification: Security and Privacy Issues - 6th International Workshop, RFIDSec 2010, Istanbul, Turkey, June 8-9, 2010, Revised Selected Papers*. 239–257. https://doi.org/10.1007/978-3-642-16822-2_19