2015 IEEE 28th Computer Security Foundations Symposium

A Complete Characterization of Secure Human-Server Communication

David Basin Saša Radomirović Michael Schläpfer*
Institute of Information Security, Department of Computer Science, ETH Zürich
Email: {david.basin, sasa.radomirovic, michael.schlaepfer}@inf.ethz.ch

***Abstract*—Establishing a secure communication channel between two parties is a nontrivial problem, especially when one or both are humans. Unlike computers, humans cannot perform strong cryptographic operations without supporting technology, yet this technology may itself be compromised. We introduce a general communication topology model to facilitate the analysis of security protocols in this setting. We use it to completely characterize all topologies that allow secure communication between a human and a remote server via a compromised computer. These topologies are relevant for a variety of applications, including online banking and Internet voting. Our characterization can serve to guide the design of novel solutions for applications and to quickly exclude proposals that cannot possibly offer secure communication.**

***Keywords*—Security Ceremonies, Formal Modeling, Security Protocols**

I. Introduction

Security-critical applications, such as online banking and Internet voting, rely on a secure communication channel between a human and a remote communication partner. These channels are constructed using security protocols that protect the messages exchanged between the human's personal computer and the remote system. However, unless the personal computer's hardware and software are trustworthy, information appearing on its screen may not faithfully represent the messages communicated with the remote system. Moreover, the personal computer may leak information to unauthorized third parties [11], [22]. Securing the last few inches of the communication channel, namely between the network cable and the human, is difficult: people need a personal computer as a communication interface, but do not want to trust it and, in contrast to computing devices, most people's computing and memorizing abilities are insufficient to perform cryptographic computations. This problem is addressed by supporting technologies, ranging from simple code sheets [8] to smart cards and hand-held readers with integrated keypads and displays, commonly used for online banking [16].

How do we formally model systems where humans, computers, and supporting technologies interact? Most existing work focuses on particular scenarios, for instance on browser-based security protocols [13], [14], login procedures [15], solutions for online banking [28], or Internet voting [24]. A general approach to modeling and reasoning about such systems are security ceremonies [10]. These extend communication protocols to include human actors and communication means that are not considered in conventional security protocol models. Security ceremonies have not been formally defined, but they have inspired a variety of formal models with different focal points, which we discuss later in Section V.

In this paper, we consider the setting of a distributed algorithm running on nodes communicating over links. We use traditional terminology and call such a distributed algorithm a protocol rather than a ceremony. We capture the abstraction of nodes communicating over links with a simple, intuitive, graph-theoretic model that we call a *communication topology*. We model the protocol execution for a given topology as a multiset term rewriting system. Our approach differs from existing approaches in that it largely ignores the interpretation of what nodes and links are and it focuses instead on their capabilities and security properties. The result is a simple and useful model with applications both to protocol verification and to establishing impossibility results.

Contributions. We introduce a communication topology model on top of an operational semantics for security protocols. Our topology model formalizes the environment in which protocols are executed and allows one to reason about communication systems at different levels of abstraction. We use the model to completely characterize necessary and sufficient conditions for the existence of security protocols that provide secure channels between a human and a remote server using an insecure network and a dishonest platform. Necessary conditions are established by impossibility results and sufficient conditions are proved constructively by providing protocols. We give proofs of our results and Tamarin models of all protocol specifications in the full version [1]. Our characterization is relevant for practical applications such as online banking and Internet voting. It allows one to quickly assess whether a particular protocol design and supporting technology can plausibly offer secure communication. The characterization can be used to guide the design of novel solutions for establishing secure channels between humans and a remote server and we provide examples that illustrate this.

Organization. We introduce our communication topology model in Section II and the underlying security protocol model in Section III. We characterize secure human-server communication in Section IV. We discuss related work in Section V, and draw conclusions in Section VI.

II. Communication Topology Model

We first define a general communication topology model that formalizes assumptions relative to which a communication protocol's security properties are analyzed. Every node in the topology corresponds to a unique role in the protocol

* Supported in part by the Swiss Federal Chancellery.

DOI 10.1109/CSF.2015.21

which specifies the node's behavior. The topology specifies the node's capabilities, initial knowledge, honesty, and available communication channels. Afterwards we restrict our focus to a particular class of topologies that is relevant for protocols where a human securely communicates with a remote server using a potentially compromised computer.

A. General Communication Topology Model

A *communication topology* (relative to two sets $NodeProp$ and $LinkProp$) is an edge- and vertex-labeled directed graph (V, E, η, μ), where V is the set of vertices, $E \subseteq V \times V$, and η and μ are functions assigning labels to vertices and edges respectively. The set of vertices V represents a protocol's roles. For $A, B \in V$, an edge $(A, B) \in E$ denotes the existence of a link from a node representing role A to the node representing role B. The vertex labeling function $\eta\colon V \to NodeProp$ assigns capability and trust assumptions to role names. It indicates, for instance, whether a role is assumed to be executed by a human and whether the executing agent is assumed to be honest. The edge labeling function $\mu\colon E \to LinkProp$ assigns channel assumptions to links, for example, whether channels are insecure, authentic, or confidential. The contents of $NodeProp$ and $LinkProp$ need not concern us now; we specify them in Section III, where our formal protocol model is defined.

We call a sequence of vertices $[v_1, \ldots, v_{n+1}] \in V^*$, such that $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq n$, a *path from* v_1 *to* v_{n+1} *of length* n or simply a *path*. The path is *acyclic* if $v_i \neq v_j$ for all $1 \leq i < j \leq n+1$. We denote the transitive closure of E by E^+, i.e., $(v_i, v_j) \in E^+$ if there is a path from v_i to v_j.

Graphical representation. We graphically represent a communication topology (V, E, η, μ) as follows. Vertices $A \in V$ are drawn as simple, concentric, or dashed circles depending on the labeling η. To express that a role $A \in V$ is assumed to be executed by a dishonest agent, we draw concentric circles. A dashed circle indicates that an honest agent executing the role A has restricted capabilities. Note that our vertex representation does not distinguish between different types of restricted capabilities and knowledge. This limitation suffices for the present paper, since humans are the only agents with restricted capabilities. Edges $e \in E$ are drawn as arrows connecting the circles and are labeled according to μ. The edge labels are written next to the arrows representing the corresponding edges.

Figure 1 shows a communication topology (V, E, η, μ), with $V = \{A, B, C, D\}$. In this example, the role A is assumed to be executed by an honest restricted agent, and role B is assumed to be executed by a dishonest agent. The remaining roles are assumed to be executed by honest, unrestricted agents. The set of edges E and their labeling can be read off of Figure 1. For example, $(A, B) \in E$, $(A, C) \notin E$, and $(A, C) \in E^+$. The link from A to D is secure ($\bullet\!\!\to\!\!\bullet$) and all other links are insecure ($\circ\!\!\to\!\!\circ$).

B. Human-Interaction Security Protocols

We now introduce the class of security protocols where humans intend to securely communicate with a remote server. We make the following assumptions regarding human capabilities.

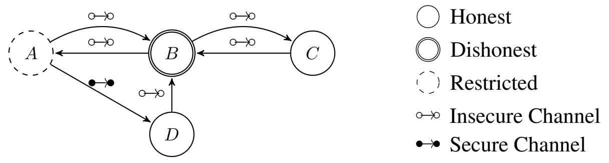


Fig. 1. Communication topology example.

Assumption 1. *Humans may send, receive, compare, concatenate (pair) and select (project) terms. They may generate random (fresh) values. No restrictions are imposed on human memory.*

Thus, humans are assumed to be able to remember all terms received on any channel[1] and to output any term constructible from their knowledge using pairing and projection on any other channel. However, they cannot perform cryptographic operations without supporting technology.

To motivate the communication topology for human-interaction security protocols, consider protocols that provide a secure communication channel between a human and a server. We can model such protocols' communication topology by defining two nodes, a human H and server S connected by a secure channel. However, this is too abstract to reason about the requirements a protocol must satisfy to provide a secure channel from the human to the server. A natural step in making this model more concrete is to assume that the human cannot directly communicate with the remote server and must instead use a computing platform P that communicates with the server over an insecure network. The resulting refined topology consists of a channel between H and P instead of H and S and an insecure channel between P and S. If we assume that the computing platform P is honest, then this topology represents the well-known problem of establishing a secure communication channel between two agents over an insecure network.

Our focus is on the case where the computing platform is dishonest, i.e., compromised. Achieving secure communication generally requires that the human has access to a trusted device D and we model this by including D in the topology. Examples of such devices are a list of one-time passwords, a code sheet, or a smart card with a corresponding card reader. Protocols that establish secure communication between the human and a remote server under these circumstances are highly relevant in practice, for example in online banking and Internet voting. We call such a protocol a *Human-Interaction Security Protocol*, or *HISP* for short, and the corresponding communication topology a *HISP topology*.

A HISP topology (formally defined in Section III-B3) consists of a human H, a server S, and a device D, which are assumed to be honest, and a computing platform P, which is assumed to be dishonest. There are no restrictions on the capabilities or initial knowledge of S, D, and P. However, H is restricted as stated in Assumption 1. Figure 2 shows the supergraph of all HISP topologies (V, E, η, μ) and indicates the edge labels. Since we use the same edge labelings in

[1]Our possibility results show that humans never need to remember more than three terms plus the names of communication partners. However, not placing limits on human memory merely strengthens our impossibility results.

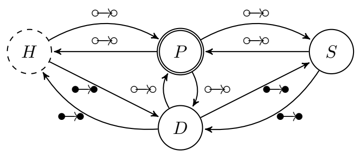


Fig. 2. The supergraph of all HISP topologies.

all HISP topologies, we often omit them from our graphical representations. Examples of such representations are shown in Section IV.

III. Security Protocol Model

In this section we describe our security protocol model which constitutes the formal underpinning of our communication topology model. Our model is based on Tamarin's [25], [19] security protocol model, which we call the *Tamarin model*. We summarize its main features and several extensions that we made to support HISPs, such as the notion of communicating knowledge. We provide further details in Appendix A. Note that although our extensions are substantial, the Tamarin tool [19], which performs deduction based on term rewriting, can still be directly applied to analyze our protocol models.

A. Background

1) Notation: We denote the set of finite sequences of elements from a set S by S^*. For the sequence s, $|s|$ denotes its length and we write s_i to refer to the i-th element of s. We write a sequence s, with $|s| = n$, as $[s_1, \ldots, s_n]$ and the empty sequence as $[\,]$. We denote the concatenation of two sequences s and s' by $s \cdot s'$. $\mathcal{P}(S)$ denotes the powerset of S.

We use the term algebra of the Tamarin model. The term algebra is denoted by $\mathcal{T}$, its underlying signature by Σ, and the set of ground terms by $\mathcal{M}$. The signature Σ contains functions $\langle \_, \_ \rangle$ for pairing, $\mathrm{senc}(\_, \_)$ and $\mathrm{sdec}(\_, \_)$ for symmetric encryption and decryption, $\mathrm{aenc}(\_, \_)$ and $\mathrm{adec}(\_, \_)$ for asymmetric encryption and decryption, $\mathrm{sign}(\_, \_)$ and $\mathrm{verify}(\_, \_, \_)$ for signing messages and verifying signatures, $\pi_1(\_)$ and $\pi_2(\_)$ for the first and second projection of a pair of terms, $\mathrm{h}(\_)$ for hashing terms, and $\mathrm{pk}(\_)$ to represent the public key corresponding to a given secret key. The function $\mathrm{pk}(\_)$ can be applied to any term t to yield the term $\mathrm{pk}(t)$, but t cannot be inferred from $\mathrm{pk}(t)$. Σ contains the two countably infinite, disjoint sets of fresh and public constants, denoted by $\mathcal{C}_{fresh}$ and $\mathcal{C}_{pub}$, respectively. Fresh constants model the generation of nonces, while public terms represent agent names and other publicly known values.

2) Multiset Term Rewriting System: We use a labeled multiset term rewriting system to represent all possible protocol behaviors. The system states are represented as finite multisets of *facts*. Facts are functions over $\mathcal{T}$ whose symbols appear in a signature Σ_{Fact} (disjoint from Σ), which is partitioned into *linear* and *persistent* fact symbols. $\mathcal{F}$ denotes the set of facts and $\mathcal{G}$ denotes the set of all ground facts, i.e., facts $\mathsf{F}(t_1, \ldots, t_n)$ such that $\mathsf{F} \in \Sigma_{Fact}$ and $t_i \in \mathcal{M}$ for all $1 \leq i \leq n$. Linear facts model resources that can only be consumed once. Persistent facts, prefixed by "!", model inexhaustible resources.

State transitions are specified by labeled multiset rewriting rules. Each such rule is denoted by $l \xrightarrow{a} r$ with $l, a, r \in \mathcal{F}^*$. The elements in l, a, r are called the rule's premises, actions, and conclusions, respectively. The transition rewrites the current state by replacing the linear facts in l with the facts in r and is labeled with the facts in a. The initial system state is the empty multiset.

A trace tr is a finite sequence of sets of actions $tr_i \in \mathcal{P}(\mathcal{G})$, for $1 \leq i \leq |tr|$. The action sets in the trace label the system's state transitions that correspond to applying a ground instance of a rule in a set $\mathcal{R}$. We write $a \in tr$ if $a \in tr_i$ for some $1 \leq i \leq |tr|$, that is, when the action a occurs in a set of ground actions in the trace tr. We denote the set of all traces for the set of rules $\mathcal{R}$ by $TR(\mathcal{R})$.

In HISP specifications we partition $\mathcal{R}$ into model rules and protocol specification rules, denoted by $\mathcal{R}_{Model}$ and $\mathcal{R}_{Spec}$ respectively. $\mathcal{R}_{Model}$ consists of: Rule (1), shown below; a fixed set of message deduction rules modeling a standard Dolev-Yao adversary [9]; and our model extensions described in Section III-B. The rules in $\mathcal{R}_{Spec}$ model a given protocol specification and are described in Section III-C.

The Tamarin rules modeling a Dolev-Yao adversary are implemented with three facts. The adversary learns all terms in Out facts and injects messages from his knowledge using In facts. Terms learned by the adversary are stored as persistent $!\mathsf{K}$ facts, which represent the adversary's knowledge. The only rule producing fresh constants and thereby creating Fr facts is

$$[\,] \rightarrow [\mathsf{Fr}(x)]. \tag{1}$$

Every fresh constant is produced at most once in a trace.

B. Model Extensions

To connect the communication topology to the underlying security protocols model, we need to define the node and link properties, i.e., the sets *NodeProp* and *LinkProp* introduced in Section II-A, in the Tamarin model.

1) Node Properties: Every node in a communication topology (V, E, η, μ) is assigned capability and trust assumptions by the vertex labeling function $\eta \colon V \rightarrow \mathit{NodeProp}$. We let $\mathit{NodeProp} = \mathcal{P}(\Sigma) \times \mathcal{P}(\mathcal{T}) \times \{\mathsf{honest}, \mathsf{dishonest}\}$. An agent's capabilities are defined by its computational abilities and initial knowledge. The computational capability assumption is specified by a subset of Σ consisting of the function symbols available to the agent executing the role that is represented by the node. The initial knowledge assumption is specified as a subset of $\mathcal{T}$. It indicates the *maximal* initial knowledge an agent is allowed to have. An empty set formalizes that the agent has no initial knowledge, while $\mathcal{T}$ states that no restrictions are placed on the agent's initial knowledge other than that it is a finite set. Note that this finite initial knowledge requirement is without loss of generality, because the initial knowledge set is not required to be closed under term inference. This is a simple way to prevent that an agent's initial knowledge contains all fresh constants. The elements in $\{\mathsf{honest}, \mathsf{dishonest}\}$ indicate the trust assumptions associated with a role. Agents marked $\mathsf{dishonest}$ are assumed to be controlled by the adversary whereas those marked honest are assumed to faithfully execute the security protocol.

$\mathcal{AG} := \{$

$$[\mathsf{Fr}(x)] \xrightarrow{\mathsf{Fresh}(A,x),\mathsf{Honest}(A)} [\mathsf{Fresh}(A,x)]\} \tag{2}$$

$$[\mathsf{AgSt}(A, step, kn)] \xrightarrow{\mathsf{Dishonest}(A)} [\mathsf{Out}(\langle A, step, kn\rangle)], \tag{3}$$

$$[\mathsf{In}(\langle step, kn\rangle)] \xrightarrow{\mathsf{Dishonest}(A)} [\mathsf{AgSt}(A, step, kn)], \tag{4}$$

$$[\mathsf{In}(x)] \xrightarrow{\mathsf{Dishonest}(A)} [\mathsf{Fresh}(A,x)] \quad \} \tag{5}$$

Fig. 3. Honest and dishonest agent rules.

We model agents explicitly with $\mathsf{AgSt}(A, step, kn)$ facts, where A is a public term representing an agent's name, $step$ refers to the role step the agent is in, and kn is the agent's knowledge at that step. The set of agents appearing in a protocol execution, denoted by $Agents(tr)$, is the set of all public constants A such that $\mathsf{AgSt}(A, step, kn)$ appears in a state of tr for some $step$ and kn. The subset of honest agents, denoted by $Honest(tr)$, is the set of all agents A such that $\mathsf{Dishonest}(A)$ does not appear in tr. We model agents with the $\mathcal{AG}$ rules shown in Figure 3. Honest agents generate fresh constants using Rule (2). These agents are marked with a Honest action. The subsequent rules concern dishonest agents. These agents are marked with a Dishonest action. By Rule (3), a dishonest agent may leak all information in its state to the adversary. Rule (4) models the adversary's capability to arbitrarily modify a dishonest agent's internal state and Rule (5) models that a dishonest agent's fresh constants may be chosen by the adversary.

2) Link Properties: Every link in a communication topology (V, E, η, μ) is assigned a channel property, representing an assumption on the link's behavior, by the edge labeling function $\mu\colon E \to LinkProp$. We define four channel properties and set $LinkProp = \{\circ\!\!\rightarrow\!\!\circ, \bullet\!\!\rightarrow\!\!\circ, \circ\!\!\rightarrow\!\!\bullet, \bullet\!\!\rightarrow\!\!\bullet\}$, where the four symbols denote the properties for insecure, authentic, confidential, and secure channels, respectively. This notation is adapted from Maurer and Schmid's channel calculus [17].

The insecure channel $\circ\!\!\rightarrow\!\!\circ$ is the standard communication channel between protocol agents in a Dolev-Yao model. We extend the Dolev-Yao message deduction rules of the Tamarin model that pertain to insecure channels with a set of channel rules, $\mathcal{CH}$, shown in Figure 4. $\mathcal{CH}$ models how protocol agents access insecure, authentic, confidential, and secure (i.e., authentic and confidential) channels. Rules (6) and (7) represent insecure channels. The sending of messages over an insecure channel is labeled with the $\mathsf{Snd_I}$ action and produces an Out fact, which represents the adversary's capability to learn messages by eavesdropping. Rule (7) is annotated with the $\mathsf{Rcv_I}$ action and represents the adversary's capability to insert arbitrary messages into insecure channels whenever a protocol agent intends to receive a message from an insecure channel (In).

The authentic channel $\bullet\!\!\rightarrow\!\!\circ$ allows the adversary to learn messages sent on the channel, but prevents the adversary from modifying the message or its sender. The adversary may, however, replay transmitted messages on this channel. Rules (8) and (9) model authentic channels. In Rule (8), the adversary learns the message (Out). The auxiliary !Auth fact ensures that in Rule (9) the adversary can neither alter the message nor its sender. The !Auth fact is persistent, which reflects the adversary's capability to replay authentically transmitted messages. The rules are annotated with the corresponding $\mathsf{Snd_A}$ and $\mathsf{Rcv_A}$ actions.

$\mathcal{CH} := \{$

$$[\mathsf{Snd_I}(A,B,m)] \xrightarrow{\mathsf{Snd_I}(A,B,m)} [\mathsf{Out}(\langle A,B,m\rangle)], \tag{6}$$

$$[\mathsf{In}(\langle A,B,m\rangle)] \xrightarrow{\mathsf{Rcv_I}(A,B,m)} [\mathsf{Rcv_I}(A,B,m)], \tag{7}$$

$$[\mathsf{Snd_A}(A,B,m)] \xrightarrow{\mathsf{Snd_A}(A,B,m)} [!\mathsf{Auth}(A,m), \mathsf{Out}(\langle A,B,m\rangle)], \tag{8}$$

$$[!\mathsf{Auth}(A,m), \mathsf{In}(B)] \xrightarrow{\mathsf{Rcv_A}(A,B,m)} [\mathsf{Rcv_A}(A,B,m)], \tag{9}$$

$$[\mathsf{Snd_C}(A,B,m)] \xrightarrow{\mathsf{Snd_C}(A,B,m)} [!\mathsf{Conf}(B,m)], \tag{10}$$

$$[!\mathsf{Conf}(B,m), \mathsf{In}(A)] \xrightarrow{\mathsf{Rcv_C}(A,B,m)} [\mathsf{Rcv_C}(A,B,m)], \tag{11}$$

$$[\mathsf{In}(\langle A,B,m\rangle)] \xrightarrow{\mathsf{Rcv_C}(A,B,m)} [\mathsf{Rcv_C}(A,B,m)], \tag{12}$$

$$[\mathsf{Snd_S}(A,B,m)] \xrightarrow{\mathsf{Snd_S}(A,B,m)} [!\mathsf{Sec}(A,B,m)], \tag{13}$$

$$[!\mathsf{Sec}(A,B,m)] \xrightarrow{\mathsf{Rcv_S}(A,B,m)} [\mathsf{Rcv_S}(A,B,m)] \quad \} \tag{14}$$

Fig. 4. Channel rules.

The confidential channel $\circ\!\!\rightarrow\!\!\bullet$ does not allow the adversary to learn the message sent on the channel, but allows the adversary to modify the sender and to repeatedly deliver (replay) the message on the confidential channel. The adversary can also deliver an arbitrary message from his knowledge (faking an arbitrary sender) on the confidential channel. Confidential channels are modeled using Rules (10)–(12). Rule (10) creates an auxiliary !Conf fact and the adversary does not learn the message. Rule (11) represents the case where the adversary passes the (unknown) confidential message m to the intended recipient, possibly pretending that it stems from another sender (In). The !Conf fact is persistent, which reflects the adversary's capability to replay confidentially transmitted messages. Rule (12) represents the adversary's capability to access the confidential channel to deliver any message from his knowledge.

Finally, for the secure channel $\bullet\!\!\rightarrow\!\!\bullet$, the adversary neither learns the message sent on it, nor can he change the sender, receiver, or transmitted message, but he may repeatedly deliver it. Rules (13) and (14) model secure channels. In Rule (13), the adversary learns nothing and an auxiliary !Sec fact is generated, which models that the adversary can neither alter the message nor its sender. Rule (14) models receiving a message from a secure channel. The !Sec fact is persistent, allowing the adversary to replay securely transmitted messages.

The protocol rules for the above channels are labeled with send and receive actions that indicate the type of channel used, the sender, receiver, and message. This means that in a protocol execution, the application of a rule that sends a message on, e.g., the authentic channel $\bullet\!\!\rightarrow\!\!\circ$ is labeled with a $\mathsf{Snd_A}(A, B, m)$ action, where A is the agent sending the message m and B is the intended recipient. The reception of a message on the confidential channel $\circ\!\!\rightarrow\!\!\bullet$ is labeled with a $\mathsf{Rcv_C}(A, B, m)$ action, where A is the apparent sender of the message m and B the recipient. The send and receive actions for the insecure and secure channels are $\mathsf{Snd_I}$, $\mathsf{Rcv_I}$ and $\mathsf{Snd_S}$, $\mathsf{Rcv_S}$, respectively. Thus every message sent or received by an agent is logged with a corresponding action in the trace.

3) HISP Topology: We can now formally define the HISP topology.

Definition 1. *A* HISP topology *is a communication topology (V, E, η, μ), where the set of nodes is $V = \{H, D, S, P\}$ and the set of links is $E \subseteq \{(a, b) \in V \times V \mid a \neq b \wedge (a, b) \neq (H, S) \wedge (a, b) \neq (S, H)\}$. The vertex labels are defined by $\eta(H) = (\Sigma_H, \mathcal{T}, \textsf{honest})$, $\eta(D) = (\Sigma, \mathcal{T}, \textsf{honest})$, $\eta(S) = (\Sigma, \mathcal{T}, \textsf{honest})$, and $\eta(P) = (\Sigma, \mathcal{T}, \textsf{dishonest})$, where $\Sigma_H = \{\langle\_,\_\rangle, \pi_1(\_), \pi_2(\_)\} \cup \mathcal{C}_{pub} \cup \mathcal{C}_{fresh}$. The edge labels are $\mu(e) = \circ\!\!\rightarrow\!\!\circ$, for $e \in E_P$, and $\mu(e) = \bullet\!\!\rightarrow\!\!\bullet$, for $e \in E \setminus E_P$, where $E_P = \{(a, b) \in E \mid a = P \vee b = P\}$.*

Example 1. *In* code voting *protocols, such as SureVote [8], code sheets assigning random codes to ballot options are distributed by the election authority to the voters prior to an election. It is assumed that the code sheets are distributed over a secure channel and that no two voters' code sheets are the same. To vote for a candidate, a voter enters the corresponding code into his untrusted computer. This code is then submitted to the election authority's server. Since the election authority created the code sheets, it can map the code back to the selected candidate.*

The HISP topology (V, E, η, μ) is shown in Figure 5. The voter H's dishonest computer P is used to submit a ballot, i.e., a candidate choice, to the election authority's server S. The pre-distributed code sheet is modeled by D. The edge

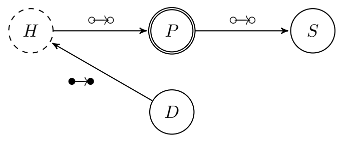


Fig. 5. HISP topology for code voting.

$(D, H) \in E$ models the voter's ability to read information from the code sheet. This communication is considered to be secure, for example the voter reads the code sheet in a private environment.

C. Protocol Specification

Protocol specification rules $l \xrightarrow{a} r \in \mathcal{R}_{Spec}$ consist of setup rules defining the roles' initial knowledge and honesty assumptions, and rules defining the message exchange steps. The setup rules must only use Fr facts in l and AgSt facts in r. This suffices to specify the initial knowledge of the protocol roles.

The message exchange rules must contain exactly one $\mathsf{AgSt}(A, \_, \_)$ fact in l, for an agent A, and may contain one or more $\mathsf{Fresh}(A, \_)$, and $\mathsf{Rcv}\_(\_, A, \_)$ facts. They may contain $\mathsf{AgSt}(A, \_, \_)$ and $\mathsf{Snd}\_(A, \_, \_)$ facts in r. This ensures that these rules can only be used for communication. Additionally, the message exchange rules may contain actions that are used to verify security properties. These actions are $\mathsf{Learn}(A, \_)$, $\mathsf{Comm}(A, \_)$, $\mathsf{Secret}(A, \_, \_)$, $\mathsf{Authentic}(\_, A, \_)$, and $\mathsf{Trust}(\_)$ and will be discussed in Section III-D. These actions must not, however, appear in the setup rules. Further protocol specification details are given in Appendix B.

For ease of reading, we represent protocols in an extended Alice & Bob notation from which the corresponding protocol rules can be easily obtained. We illustrate this on an example below. The extension of the Alice & Bob notation contains the symbols in the *LinkProp* set. For instance, we write $A \circ\!\!\rightarrow\!\!\circ B\colon m$ to express that a message m is to be sent from an agent executing role A to an agent executing role B over an insecure channel. To express that the message is sent over an authentic channel, we write $A \bullet\!\!\rightarrow\!\!\circ B\colon m$.

To specify the initial knowledge m of an agent executing role A, we write $A\colon \mathrm{knows}(m)$. To express that the agent generates fresh constants $m_1, \ldots, m_n$, we write $A\colon \mathrm{fresh}(m_1, \ldots, m_n)$ or $A \circ\!\!\rightarrow\!\!\circ B\colon \mathrm{fresh}(m_1, \ldots, m_n).m$ when the generation is followed by a send event.

In general, an Alice & Bob specification leaves room for different interpretations [5]. When such ambiguities arise, we indicate both the message sent and the message pattern expected to be received and separate them with " / ", as in $A \circ\!\!\rightarrow\!\!\circ B\colon m \;/\; m'$. The variables in m' determine how the received message is parsed by an agent executing the role B.

Example 2. *In the code voting protocol of Example 1, the voter H possesses a personal code sheet D. The latter contains the candidate names and corresponding codes, bound to H and S. The election server S is initialized to know the distributed code sheet D, the candidate names and the corresponding codes as well as H to whom the code sheet was distributed. Voter H first reads the tuple $\langle cand, c\rangle$ from the code sheet, where $cand$ represents the desired candidate and c the corresponding code. Using his dishonest computer P, H submits c to S. The election authority maps c back to $cand$. The protocol is specified in Alice & Bob notation as shown in Figure 6. The corresponding protocol rules $\mathcal{R}_{CodeVoting}$*

$$\begin{aligned}
D&\colon \mathrm{knows}(\langle H, S, cand, c\rangle)\\
S&\colon \mathrm{knows}(\langle H, D, cand, c\rangle)\\
D \bullet\!\!\rightarrow\!\!\bullet H&\colon \langle S, cand, c\rangle\\
H \circ\!\!\rightarrow\!\!\circ P&\colon c\\
P \circ\!\!\rightarrow\!\!\circ S&\colon c
\end{aligned}$$

Fig. 6. Code Voting Protocol

are shown in Figure 7. The initial knowledge specified in the Alice & Bob specification is set up in Rule (15)*. Each of the three communication steps is specified by two rules: one for the sender followed by one for the receiver. The rules consume and produce* AgSt *facts that contain the agent's knowledge and keep track of the agent's protocol execution. The term ε denotes that the agent has no knowledge. Quoted strings, such as 'D_0', are elements of $\mathcal{C}_{pub}$ and are used to denote the agents' protocol steps. The rules produce* Snd *facts for sending messages and consume* Rcv *facts for the received messages. The former are transformed into the latter by the channel rules shown in Figure 4 and correspond to the link properties specified in the Alice & Bob specification. Rules* (18) *and* (21) *contain actions that are related to the protocol's security claims and are defined in Section III-D. The complete protocol specification in our model is then given by $\mathcal{R}_{CodeVoting} \cup \mathcal{R}_{Model}$, where $\mathcal{R}_{Model}$ contains Rule* (1)*, the*

$$\begin{aligned}
&\mathcal{R}_{\text{CodeVoting}} = \big\{ \\
&[\mathsf{Fr}(cand), \mathsf{Fr}(c)] \rightarrow [\mathsf{AgSt}(D, \text{'}D_0\text{'}, \langle H, S, cand, c\rangle), \mathsf{AgSt}(H, \text{'}H_0\text{'}, \varepsilon), \mathsf{AgSt}(P, \text{'}P_0\text{'}, \varepsilon), \mathsf{AgSt}(S, \text{'}S_0\text{'}, \langle H, D, cand, c\rangle)] && (15)\\
&[\mathsf{AgSt}(D, \text{'}D_0\text{'}, \langle H, S, cand, c\rangle)] \rightarrow [\mathsf{Snd}_{\mathsf{S}}(D, H, \langle S, cand, c\rangle), \mathsf{AgSt}(D, \text{'}D_1\text{'}, \langle H, S, cand, c\rangle)] && (16)\\
&[\mathsf{AgSt}(H, \text{'}H_0\text{'}, \varepsilon), \mathsf{Rcv}_{\mathsf{S}}(D, H, \langle S, cand, c\rangle)] \rightarrow [\mathsf{AgSt}(H, \text{'}H_1\text{'}, \langle D, S, cand, c\rangle)] && (17)\\
&[\mathsf{AgSt}(H, \text{'}H_1\text{'}, \langle D, S, cand, c\rangle)] \xrightarrow{[\mathsf{Comm}(H,cand), \mathsf{Secret}(H,S,cand), \mathsf{Trust}(D)]} [\mathsf{Snd}_{\mathsf{I}}(H, P, c), \mathsf{AgSt}(H, \text{'}H_2\text{'}, \langle D, S, cand, c\rangle)] && (18)\\
&[\mathsf{AgSt}(P, \text{'}P_0\text{'}, \varepsilon), \mathsf{Rcv}_{\mathsf{I}}(H, P, c)] \rightarrow [\mathsf{AgSt}(P, \text{'}P_1\text{'}, c)] && (19)\\
&[\mathsf{AgSt}(P, \text{'}P_1\text{'}, c)] \rightarrow [\mathsf{Snd}_{\mathsf{I}}(P, S, c), \mathsf{AgSt}(P, \text{'}P_2\text{'}, c)] && (20)\\
&[\mathsf{AgSt}(S, \text{'}S_0\text{'}, \langle H, D, cand, c\rangle), \mathsf{Rcv}_{\mathsf{I}}(P, S, c)] \xrightarrow{[\mathsf{Learn}(S,cand), \mathsf{Authentic}(H,S,cand), \mathsf{Trust}(D)]} [\mathsf{AgSt}(S, \text{'}S_1\text{'}, \langle H, D, cand, c\rangle)] \quad \big\} && (21)
\end{aligned}$$

Fig. 7. HISP specification of the code voting protocol.

sets $\mathcal{AG}$ and $\mathcal{CH}$, defined in Section III-B, as well as standard rules governing the adversary's message deduction capability, which are discussed in Appendix A.

D. Channels as Goals

In Section III-B2, we defined communication channels as a means for agents to communicate. Here we define the notion of a communication channel as a protocol goal. This provides us with a formal meaning for statements asserting the existence or non-existence of protocols providing secure channels in HISP topologies. The alignment of the semantics of our HISP model with the semantics of Tamarin is particularly significant here because it allows us to give manual proofs of impossibility results and use the Tamarin tool to obtain automatic proofs of possibility results in the same protocol model. In particular, our possibility results are proven for unbounded numbers of interleaved protocol sessions and remain true for all equational theories supported by Tamarin that include the standard theory used here.

Our use of channels as goals has three aspects we highlight here. First, we consider the *communication of knowledge* rather than just the transmission of messages over a network. We formally define this concept in Definition 3 and illustrate its application thereafter. Second, to avoid protocols that trivially satisfy security properties by never communicating a useful message, we require that there exists a trace in which security-relevant knowledge is communicated from one honest agent to another. We therefore define the notion of *providing a communication channel*. Finally, we consider as a special case protocols in which a fresh constant generated by the sender can be communicated. Such protocols are said to provide an *originating communication channel*. We use this as a coarse, but for our purposes sufficient, way to differentiate between protocols that allow for the communication of an arbitrary message and protocols that impose limits on the communicated message, such as that it be a yes/no vote.

We first define what it means for a protocol to provide a particular type of channel. A channel property is a pair of predicates (p, q), each of which has domain $\mathcal{P}(\mathcal{G})^* \times \mathcal{C}_{pub} \times \mathcal{C}_{pub} \times \mathcal{M}$. A protocol provides a channel with a property defined by (p, q) if (1) there exists a trace, two honest agents, and a message, such that p is satisfied and (2) for all traces, agents, and messages, q is satisfied. The existential requirement p ensures that the protocol provides some given functionality, such as communicating messages. The universal requirement q specifies a safety property, such as confidentiality. In order to reason about the (im-)possibility of secure communication, we need both of these requirements.

Definition 2. *Protocol $\mathcal{R}$* provides a channel with the property *(p, q) if*

$$\exists\ tr \in TR(\mathcal{R}), S, R \in \mathit{Honest}(tr), m \in \mathcal{M}\colon p(tr, S, R, m) \wedge$$
$$\forall\ tr \in TR(\mathcal{R}), S, R \in \mathit{Honest}(tr), m \in \mathcal{M}\colon q(tr, S, R, m).$$

We now define several channel properties, starting with the properties related to communication of knowledge and origination and concluding with security properties.

We define what it means for knowledge to be communicated as follows. We say that an agent S communicates a message m in a trace, if the action $\mathsf{Comm}(S, m)$ appears in the trace. This merely implies that S knows m, but there is no guarantee that m is sent on the network. We say that an agent R learns a message m in a trace, if $\mathsf{Learn}(R, m)$ appears in the trace. This too implies that R knows m, but there is no guarantee that R did not know m earlier in the trace. To say that m is communicated from S to R in a trace means that $\mathsf{Comm}(S, m)$ occurs before $\mathsf{Learn}(R, m)$ in the trace. In other words, the agents S and R know m and S performs a protocol step labeled $\mathsf{Comm}(S, m)$ before R performs a protocol step labeled $\mathsf{Learn}(R, m)$.

Definition 3. *A message $m \in \mathcal{M}$ is said to be* communicated *from an agent S to an agent R in a trace tr, denoted* $\text{communicate}(tr, S, R, m)$*, if*

$$\begin{aligned}\exists\ tr', tr'' \in \mathcal{P}(\mathcal{G})^* : tr = tr' \cdot tr''\\ \wedge\ \mathsf{Comm}(S, m) \in tr' \wedge \mathsf{Learn}(R, m) \in tr''.\end{aligned}$$

A communication channel *is defined by the property (p_{com}, q_{com}), where*

$$p_{com}(tr, S, R, m) := \text{communicate}(tr, S, R, m),$$
$$q_{com}(tr, S, R, m) := \top.$$

Note that in the definition above, the predicate $\top$ (true) places no additional requirement on the set of traces. We say that a protocol *provides a communication channel* if the protocol satisfies the communication channel property. Intuitively, this states that the protocol is indeed a functioning communication protocol: it allows an honest agent to communicate a message to another honest agent. We will use analogous terminology for the channel properties to be defined in the remainder of this section.

Remark 1. *For a protocol to provide a communication channel, there must be a trace in which the* $\mathsf{Comm}(S, m)$ *and* $\mathsf{Learn}(S, m)$ *actions occur in the given order. These occurrences may, however, be coincidental. The requirement that these actions are appropriately ordered in all traces is given by the authenticity property below (Definition 6). The purpose of the communication channel property is to ensure the* possibility *that a protocol can transfer knowledge from one agent to another. This weak condition combined with, e.g., a confidentiality requirement, ensures that a protocol does not trivially satisfy the confidentiality requirement by not transferring any knowledge.*

Note also that communicating a message from an agent S to an agent R is more general than transmitting a message from S to R. If R receives a message m from S, then S has communicated m to R. However, a message can be communicated without being sent, as the next example shows.

Example 3. *Consider the code voting protocol of Example 2. The human* H *communicates the candidate* $cand$ *to the voting server* S *by sending the code* c*. This is expressed in Rule* (18) *of Figure 7 with the actions* $\mathsf{Comm}(H, cand)$*. When this rule is applied in a protocol execution, the* $\mathsf{Snd_I}(H, P, c)$ *fact is produced and the action* $\mathsf{Comm}(H, cand)$ *occurs in the trace. Thus the message* c *is sent over the network. When Rule* (21) *is applied, the* $\mathsf{Rcv_I}(P, S, c)$ *fact is consumed, thus the server receives* c *from the network and* $\mathsf{Learn}(S, cand)$ *occurs in the trace, and thus* $cand$ *is learned by* S*. This is a valid step because* S *has the pair* $(cand, c)$ *in its knowledge as seen by the* $\mathsf{AgSt}(S, \text{'}S_0\text{'}, \langle H, D, cand, c\rangle)$ *fact, which is consumed by the same rule.*

A protocol where the sender communicates a message by sending its code limits the sender's communication channel to the messages on the code sheet. This is useful for applications like code voting, but cumbersome for an email application where senders communicate arbitrary messages. For email, the shared code sheet would be better used to establish a shared cryptographic key for securing subsequent email communication. This, however, is a different protocol and is not an option for humans who cannot perform encryption without supporting technology. For this reason we define the *originating channel* property to make the fundamental distinction between protocols that allow for the communication of a fresh constant generated by the sender and those that do not. An originating channel represents the ability to *generate* an arbitrary message.

Definition 4. *We say that a message* m originates *with an agent* A *in a trace* tr*, if* m *is a fresh term that* A *generates, that is, if* $\mathsf{Fresh}(A, m) \in tr$*. An* originating channel *is defined by the property* (p_{orig}, q_{orig})*, where*

$$p_{orig}(tr, S, R, m) := \mathsf{Fresh}(S, m) \in tr,$$
$$q_{orig}(tr, S, R, m) := \top.$$

A protocol providing an originating channel allows agents to generate fresh constants. The only protocol rule in our model that has a $\mathsf{Fresh}(A, x)$ action is Rule (2). It is also the only rule that allows honest agents to generate a fresh constant.

Remark 2. *Non-originating channels are not limited to public constants. A channel is non-originating for an agent if the agent does not* generate *a fresh constant. This does not exclude the use of fresh constants that the agent receives from another agent, reads from code-sheets, or that are in the agent's initial knowledge.*

We use the originating channel property together with the communication channel property to model an agent's ability to communicate an arbitrary message. For instance, an email protocol must provide a communication channel and an originating channel with respect to the same message m.

We say that a protocol *combines* channel properties (p_1, q_1) and (p_2, q_2) if it satisfies the property $(p_1 \wedge p_2, q_1 \wedge q_2)$. In this case, we combine the adjectives used to describe the channel properties. For instance, we say that a protocol *provides an originating communication channel* if it combines an originating channel with a communication channel.

Example 4. *The code voting protocol of Example 2 provides a communication channel from* H *to* S *because there is a trace* tr *satisfying* $communicate(tr, H, S, cand)$*. The trace is obtained by applying Rule* (1) *twice, followed by the Rules* (15) *through* (21)*, in that order, except that they are interleaved with the Channel Rules* (13) *and* (14) *to transform the* $\mathsf{Snd_S}$ *into the* $\mathsf{Rcv_S}$ *fact and Rules* (6) *and* (7) *to transform the two* $\mathsf{Snd_I}$ *into the two* $\mathsf{Rcv_I}$ *facts. The protocol does not provide an originating communication channel, because there is no trace* tr *for which the* $communicate(tr, H, S, cand)$ *predicate holds and in which the* $\mathsf{Fresh}(H, cand)$ *action is produced.*

Remark 3. *If a message that originates with an agent (e.g. an email) can be encoded as a sequence of (non-originating) code-words, then non-originating channels can be used repeatedly to transmit the code-words. This means that the protocol providing a non-originating channel must be executed repeatedly. The number of repetitions depends on the number of code-words that are needed the encode the message. In contrast, a protocol providing an originating channel need only be executed once to communicate the entire message. We capture this difference in our symbolic model by distinguishing between these two types of channels. This distinction is natural for the HISP setting: For humans, the repeated execution of a protocol to encode an arbitrary message by many code-words is theoretically possible, but inconvenient and unrealistic in practice, except in isolated military contexts.*

The communication channel and originating channel properties defined above concern protocols' functionality. We now define confidentiality and authenticity of messages, which are safety properties. A channel has the confidentiality property if the adversary does not learn a specified message. To identify the messages m that should remain confidential in a protocol, we annotate a protocol rule with a $\mathsf{Secret}(S, R, m)$ action.

Definition 5. *The* confidentiality property *is defined by* (p_{conf}, q_{conf})*, where*

$$p_{conf}(tr, S, R, m) := \mathsf{Secret}(S, R, m) \in tr$$
$$q_{conf}(tr, S, R, m) := \mathsf{Secret}(S, R, m) \in tr \rightarrow !\mathsf{K}(m) \notin tr.$$

A channel has the authenticity property for the agents S and R, if whenever R learns m, then m was previously communicated by S. To specify that a message m should be authentically communicated in a protocol, we annotate the protocol rule in which the message is learned with an $\mathsf{Authentic}(S, R, m)$ action.

Definition 6. *The* authenticity property *is defined by* (p_{auth}, q_{auth})*, where*

$$\begin{aligned} p_{auth}(tr, S, R, m) &:= \mathsf{Authentic}(S, R, m) \in tr \\ q_{auth}(tr, S, R, m) &:= \mathsf{Authentic}(S, R, m) \in tr \\ &\qquad \rightarrow \mathrm{communicate}(tr, S, R, m). \end{aligned}$$

We call the combination of a confidential channel and an authentic channel a *secure* channel.

Remark 4. *We will henceforth only consider protocols that provide a communication channel combined with other channel properties. We will therefore omit the word "communication" for the channels provided by the protocols.*

Additional Channel Properties. One contribution of our work is to characterize the settings in which secure communication channels exist, even when some communication partners are dishonest. We therefore must explicitly state which roles of a protocol are assumed to be executed by honest agents. This is done by annotating a protocol rule with the action $\mathsf{Trust}(A)$, where A is an agent.

We distinguish between the trust assumptions for confidentiality and authenticity and therefore define two properties.

Definition 7. *The* trust assumption for confidentiality *is defined by the property* (p_{ctrust}, q_{ctrust})*, where*

$$\begin{aligned} p_{ctrust}(tr, S, R, m) &:= \exists\, T \in \mathcal{C}_{pub}, i \in \{1, \dots, |tr|\} : \\ &\qquad \mathsf{Trust}(T) \in tr_i \wedge \mathsf{Secret}(S, R, m) \in tr_i \\ &\qquad \wedge\, T \in \mathit{Honest}(tr) \\ q_{ctrust}(tr, S, R, m) &:= \forall\, T \in \mathcal{C}_{pub}, i \in \{1, \dots, |tr|\} : \\ &\qquad \mathsf{Trust}(T) \in tr_i \wedge \mathsf{Secret}(S, R, m) \in tr_i \\ &\qquad \rightarrow T \in \mathit{Honest}(tr). \end{aligned}$$

The trust assumption for authenticity *is defined by the property* (p_{atrust}, q_{atrust}) *which is identical to the property* (p_{ctrust}, q_{ctrust}) *except for the action* $\mathsf{Authentic}(S, R, m)$ *in place of the action* $\mathsf{Secret}(S, R, m)$.

The two properties state that if a $\mathsf{Secret}(S, R, m)$ or $\mathsf{Authentic}(S, R, m)$ action occurs with a $\mathsf{Trust}(T)$ action, then the agent T is honest. We can use these properties to state that whenever a confidentiality or authenticity claim is made, the specified intended communication partners are assumed to be honest. We achieve this statement with a relativization.

We say that a protocol provides the channel property (p_1, q_1) relative to the channel property (p_2, q_2) if it satisfies the property $(p_1 \wedge p_2, q_1 \vee \neg q_2)$. That is, both existential predicates must be satisfied, and the universal predicate q_2 implies q_1. For instance, the property $(p_{\mathrm{conf}} \wedge p_{\mathrm{ctrust}}, q_{\mathrm{conf}} \vee \neg q_{\mathrm{ctrust}})$ specifies that a protocol provides a confidential channel *if* the sender's trusted communication partners are honest.

Example 5. *To verify that the code voting protocol from Example 2 provides a confidential channel from H to S in the HISP topology shown in Example 1, we must verify the property* $(p_1, q_1) = (p_{com} \wedge p_{conf} \wedge p_{ctrust}, (q_{com} \wedge q_{conf}) \vee \neg q_{ctrust})$. *If we let (p_2, q_2) denote the analogously defined property for the authentic channel from H to S, then $(p_1 \wedge p_2, q_1 \wedge q_2)$ is the property that must be satisfied for the protocol to provide a secure channel.*

Note that we must also verify that the protocol is a valid protocol for the HISP topology. This entails checking that (1) all channels specified in the protocol $\mathcal{R}_{CodeVoting}$ are present in the HISP topology, (2) that the specified channel properties match the topology's corresponding link properties, and (3) that the specified roles satisfy the corresponding node properties. These are simple checks. First, in Rule (18) *of the code voting protocol, the action facts* $\mathsf{Comm}(H, cand)$, $\mathsf{Secret}(H, S, cand)$*, and* $\mathsf{Trust}(D)$ *indicate that the agents H, S, and D in the protocol specification correspond, respectively, to the roles H, S, and D in the HISP topology.*

We verify that the agents D and H communicate via $\mathsf{Snd_S}$ *and* $\mathsf{Rcv_S}$ *facts matching the link label $\mu((D, H)) = \bullet\!\!\rightarrow\!\!\bullet$ in the topology. The remaining channel facts (1) are verified analogously.*

To verify the node properties (2), we note that message derivations specified for H involve only pairing and projection.

Finally, we verify that the trust assumptions (3) are correct: P is the only agent marked dishonest in the HISP topology, thus we must verify that agent P in the protocol specification makes no security claims and is not indicated to be trusted. Indeed, none of the actions in Rules (18) *and* (21) *contain P as an argument.*

IV. Complete Classification of HISPs

The objective of a HISP is to provide a secure channel from the human H to the server S or vice versa. In this section we provide a complete classification of which HISP topologies allow such protocols. We first prove two general impossibility results concerning the establishment of confidential and authentic channels between agents. Then we classify the HISP topologies for which protocols exist that provide an originating secure channel. Such protocols permit the communication partners to securely exchange arbitrary messages. Afterwards, we consider the general case of HISPs that provide secure channels.

A. General Impossibility Results

The following two lemmas are impossibility results for secret establishment when confidential or authentic channels are available. They can be considered folklore, although, to the best of our knowledge, there are no published proofs for their statements. Impossibility results for secret establishment over insecure channels have been proven by Schmidt et al. [26].

The first lemma states the topological conditions under which no confidential channel from an honest agent S to an honest agent R can be created: If one of the agents has no initial knowledge, then there is no protocol that provides a confidential channel from S to R, even if S may send messages via authentic channels to R and R may send messages via confidential channels to S.

Lemma 1. *Let $\tau = (V, E, \eta, \mu)$ be a communication topology where $S, R \in V$ are distinct roles such that $\eta(S) =$*

$(\Sigma_S, K_S, \textsf{honest})$, $\eta(R) = (\Sigma_R, K_R, \textsf{honest})$ and $K_S = \emptyset$ or $K_R = \emptyset$. If the following two conditions are satisfied, then there exists no protocol for τ that provides a confidential channel from S to R.

1) $\forall (a, b) \in E : a \neq b \wedge (a = S \vee b = R)$
$\rightarrow \mu(a, b) \in \{\circ\!\!\rightarrow\!\!\circ, \bullet\!\!\rightarrow\!\!\circ\}$
2) $\forall (a, b) \in E : a \neq b \wedge (a = R \vee b = S)$
$\rightarrow \mu(a, b) \in \{\circ\!\!\rightarrow\!\!\circ, \circ\!\!\rightarrow\!\!\bullet\}$

To prove Lemma 1, we map every trace where a message is confidentially communicated from S to R to a trace where S performs the same protocol steps, yet the adversary learns the message by impersonating R to S. This is possible because the messages from R to S are not authenticated. Thus, S cannot distinguish between information that R sends to S and information that the adversary sends. The technical details are given in the full version [1].

The following lemma states the dual of the preceding one: If an honest agent S has no access to an authentic (or secure) channel and another honest agent R has no access to a confidential (or secure) channel, then there is no protocol that provides an authentic channel from S to R.

Lemma 2. *Let $\tau = (V, E, \eta, \mu)$ be a communication topology where $S, R \in V$ are distinct roles such that $\eta(S) = (\Sigma_S, K_S, \textsf{honest})$, $\eta(R) = (\Sigma_R, K_R, \textsf{honest})$ and $K_S = \emptyset$ or $K_R = \emptyset$. If the following two conditions are satisfied, then there exists no protocol for τ that provides an authentic channel from S to R.*

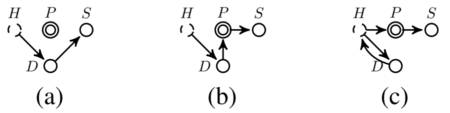
1) $\forall (a, b) \in E : a \neq b \wedge (a = S \vee b = R)$
$\rightarrow \mu(a, b) \in \{\circ\!\!\rightarrow\!\!\circ, \circ\!\!\rightarrow\!\!\bullet\}$
2) $\forall (a, b) \in E : a \neq b \wedge (a = R \vee b = S)$
$\rightarrow \mu(a, b) \in \{\circ\!\!\rightarrow\!\!\circ, \bullet\!\!\rightarrow\!\!\circ\}$

Note that we can strengthen Lemmas 1 and 2 by relaxing the empty initial knowledge condition on the agents. Instead of requiring that one of the two agents S and R has an empty initial knowledge, it suffices to make a restriction on terms that contain fresh constants. More precisely, for one of the two agents, say R, any fresh constant x occurring as a subterm of a term in the initial knowledge of R is either known to the adversary or no agent other than R has a term in his initial knowledge that contains x as a subterm.

B. Originating Secure Channels

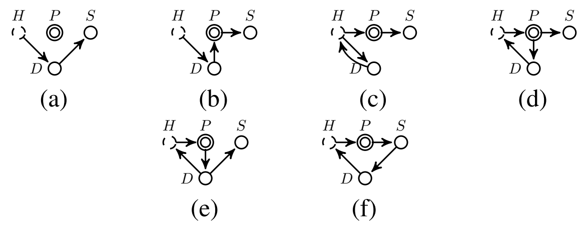
For a human to send an arbitrary message securely to a remote server, we expect that the message must be input into a trusted device. To prove this, we separate the secure channel into its confidential and authentic components. There are no surprises for the confidential channel: A human can send a confidential message to a server if and only if the human can input the message into a trusted device and there is a communication path from the trusted device to the server.

Theorem 1. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology. There exists a protocol for τ that provides an originating confidential channel from H to S if and only if $(H, D) \in E$ and $(D, S) \in E^+$. The following are all minimal graphs satisfying these conditions.*



Perhaps surprisingly, the possibilities for originating authentic channels are less restrictive than for originating confidential channels. As we now show, there are originating authentic channels from a human to a server, where the human *receives* a message from the trusted device instead of inputting one into it.

Theorem 2. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology. Then there exists a protocol for τ that provides an originating authentic channel from H to S if and only if $(H, S) \in E^+$, there exists an edge between H and D, and there exists an edge incoming to D as well as an edge outgoing from D. The following are all minimal graphs satisfying these conditions.*



The difference between the two theorems reflects the human's limitations. The human's ability to generate fresh messages and compare previously sent messages with received messages suffices to guarantee originating authenticity for certain HISP topologies, but it is insufficient for originating confidentiality. The following example illustrates this difference.

Example 6. *Let $\tau = (V, E, \eta, \mu)$ be the HISP topology shown in Figure 8 for the following scenario. A human user has a device with a small display. This is represented by $(D, H) \in E$ in τ. The device is connected to and receives input from the user's computer, so $(P, D) \in E$. The user sends messages to the server through the computer, therefore $(H, P) \in E$ and $(P, S) \in E$.*

Figure 9 presents a protocol for this HISP topology that provides an originating authentic, but not confidential, channel from the human user to the remote server. Namely, the user inputs his message m into the computer, which forwards it to the device. The device displays m along with a message authentication code (represented as a keyed hash) to the user. The message authentication code is computed by the device using a symmetric key k_{DS} that the device shares with the remote server. The user inputs the code mac into the computer, which sends the message along with the code to the remote server. The correctness of the originating authenticity claim is verified by Tamarin. Hence the protocol provides an originating authentic channel as our model is faithfully represented by multiset rewriting rules within Tamarin. Since the graph shown in Figure 8 is not a supergraph of any of the graphs shown in Theorems 1 and 3, there is no protocol for this topology that provides a confidential channel in either direction.

Combining Theorems 1 and 2 shows that the topology of

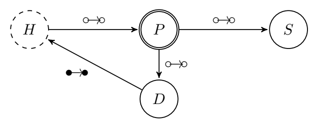


Fig. 8. HISP topology.

$$\begin{aligned}
D &: \text{knows}(\langle H, S, k_{DS}\rangle)\\
S &: \text{knows}(\langle H, D, k_{DS}\rangle)\\
H \circ\!\!\rightarrow\!\!\circ\, P &: \text{fresh}(m).m\\
P \circ\!\!\rightarrow\!\!\circ\, D &: m\\
D \bullet\!\!\rightarrow\!\!\bullet\, H &: \langle m, h(\langle k_{DS}, m\rangle)\rangle \;/\; \langle m, mac\rangle\\
H \circ\!\!\rightarrow\!\!\circ\, P &: mac\\
P \circ\!\!\rightarrow\!\!\circ\, S &: \langle m, mac\rangle \;/\; \langle m, h(\langle k_{DS}, m\rangle)\rangle
\end{aligned}$$

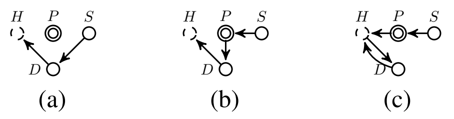
Fig. 9. A protocol providing an originating authentic channel from H to S.

any HISP providing an originating secure channel from H to S is a supergraph of one of the graphs shown in Theorem 1.

Corollary 1. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology. There exists a protocol for τ that provides an originating secure channel from H to S if and only if $(H, D) \in E$ and $(D, S) \in E^+$.*

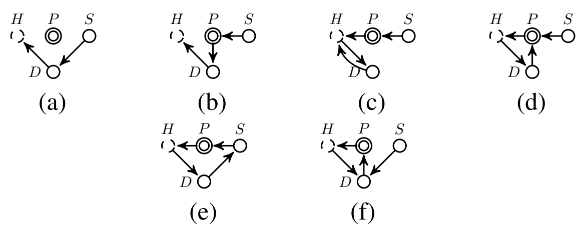
A similar situation arises in the reverse direction. An originating confidential channel from the server to the human requires that the human receives the server's message from the trusted device.

Theorem 3. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology. There exists a protocol for τ that provides an originating confidential channel from S to H if and only if $(D, H) \in E$ and $(S, D) \in E^+$. The following are all minimal graphs satisfying these conditions.*



Analogous to Theorem 2, the conditions for a human to receive an originating authentic message from a server are weaker than the conditions for originating confidential messages.

Theorem 4. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology. Then there exists a protocol for τ that provides an originating authentic channel from S to H if and only if $(S, H) \in E^+$, there exists an edge between H and D, and there exists an edge incoming to D as well as an edge outgoing from D. The following are all minimal graphs satisfying these conditions.*



Theorems 3 and 4 imply Corollary 2, which states that the topology of any HISP that provides an originating secure channel from S to H is a supergraph of one of three graphs shown in Theorem 3.

Corollary 2. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology. There exists a protocol for τ that provides an originating secure channel from S to H if and only if $(D, H) \in E$ and $(S, D) \in E^+$.*

Note that a closer inspection of the proofs of the results in this section shows that all four theorems hold even if no initial knowledge is given to the human H. This can be seen by inspecting the protocols used to prove the possibility results [1].

C. Secure Channels

In this section we classify all HISP topologies for which there exist protocols that provide secure channels. As opposed to HISPs that provide originating secure channels, these protocols may restrict the communication partners to a predefined set of messages that can be securely exchanged, such as codewords for candidates in an Internet voting system. Due to the weaker requirements regarding the origin of the exchanged messages, the set of HISP topologies for which protocols exist providing a secure channel is a superset of the former set of topologies. In the following example we sketch a HISP that provides a secure channel but not an originating secure channel.

Example 7. *Suppose the human H needs to receive the result of a medical test from a testing facility S. As this information is sensitive, the human's computing platform P must not learn or modify this information. There are only few possible test outcomes and the result can therefore be communicated to H over a non-originating channel. To this end, H generates for each possible outcome a random code word. Then H uses a trusted device D to securely transmit the outcome/code word pairs to S. Once the test result is available, S sends to H via P the code word corresponding to the test result. Thus P receives a code word, but does not learn the corresponding test result. Since P does not know the other code words, it cannot change the result. The channel is non-originating, since S cannot communicate an arbitrary message to H, but only the code words selected by H. This initial sketch of this HISP can now be specified in detail and verified with Tamarin.*

This example illustrates how our topology model and characterization can systematically guide us to HISPs. We discuss this design process in Example 8 after presenting the characterization of the available HISP topologies.

We now classify all HISPs with respect to protocols providing secure channels from H to S and vice versa. We first consider the case where H has no initial knowledge and then discuss shared knowledge. Our main results are stated in the following two theorems. Theorem 5 shows the four minimal HISP topologies for which a protocol exists that provides a secure channel from a human H to a server S.

Theorem 5. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology where $K_H = \emptyset$. Then there is a protocol for τ that provides a secure channel from H to S if and only if τ either contains an edge from D to H and a path from H to S or contains an edge*

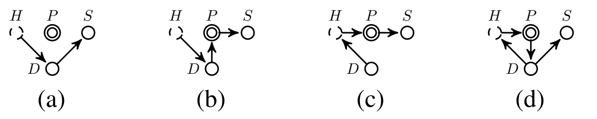
Condition	Authentic	Confidential
$(D,H) \notin E$ $\wedge(H,D) \notin E$	no, by Lemma 3	no, by Lemma 3
$(D,H) \notin E$ $\wedge(H,D) \in E$ $\wedge(D,S) \notin E^+$	no, by Lemma 4	no, by Lemma 4
$(D,H) \notin E$ $\wedge(H,D) \in E$ $\wedge(D,S) \in E^+$	yes, by Lemma 6	yes, by Lemma 6
$(D,H) \in E$	yes, by Lemma 8	yes, by Lemma 8

TABLE I. CLASSIFICATION OF ALL HISP TOPOLOGIES THAT CONTAIN A PATH FROM H TO S.

Condition	Authentic	Confidential
$(D,H) \notin E$ $\wedge(H,D) \notin E$	no, by Lemma 3	no, by Lemma 3
$(D,H) \notin E$ $\wedge(H,D) \in E$ $\wedge(D,H) \notin E^+$	no, by Lemma 5	no, by Lemma 5
$(D,H) \notin E$ $\wedge(H,D) \in E$ $\wedge(D,H) \in E^+$	yes, by Lemma 9	iff $(H,S) \in E^+$ by Lemma 10
$(D,H) \in E$	yes, by Lemma 7	yes, by Lemma 7

TABLE II. CLASSIFICATION OF ALL HISP TOPOLOGIES THAT CONTAIN A PATH FROM S TO H.

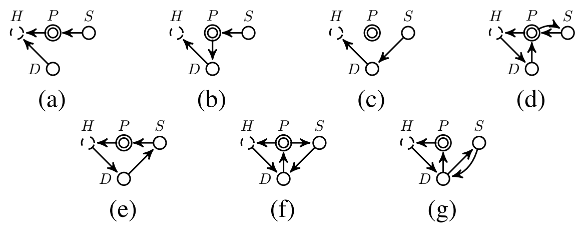
from H to D and a path from D to S. All minimal graphs satisfying these conditions are shown below.



Proof: We prove the theorem by case distinction. Table I classifies all HISP topologies that contain a path from H to S. For all other topologies, no protocol that provides an authentic, confidential, or secure channel from H to S can exist, because no information can be communicated from H to S. The cells state whether protocols providing authentic or confidential channels from H to S exist under the conditions shown in the first column. These statements are proven by the lemmas referenced in the table. The statements of Lemmas 3 through 10 are given in Appendices C and D. Proof details are given in the full version [1]. ■

Theorem 6 shows the seven minimal HISP topologies for which a protocol exists that provides a secure channel from a server S to a human H. Its proof is analogous to the proof of Theorem 5 and follows from Table II and the lemmas referenced therein.

Theorem 6. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology where $K_H = \emptyset$. Then there is a protocol for τ that provides a secure channel from S to H if and only if τ either contains an edge from D to H and a path from S to H or τ contains an edge from H to D and a path from S to itself that includes D and H. All minimal graphs satisfying these conditions are shown below.*



In the following example, we show how the minimal topologies of Theorem 6 can guide the design of a protocol that provides a secure channel from S to H.

Example 8. *We return to the scenario of Example 7 where medical test results should be securely communicated from S to H. We are interested in a protocol where H can suggest code words to be used for the test results. It follows from our characterization that this requires a path from H to S and excludes protocols based on the topologies (a)–(c).*

Topology (d) suggests a protocol where H enters outcome/code word pairs into a device D that is connected to P. The code words are signed and encrypted by D and sent to S via P. The code word corresponding to the test result is sent from S to P, which displays it to H.

Topology (e) has the simplest protocol flow. If we assume that postal mail is secure and that the medical test is a mail-in test, then the topology suggests that D could be a paper form provided with the test kit. The human fills in the form with code words next to the possible test outcomes and sends it with the kit to the testing facility. The resulting code word is communicated back to the human as above.

Topologies (f) and (g) apply in a scenario where the testing facility provides electronic data, but does not operate a download server. The protocol starts identically to the one outlined for topology (d). The results are sent back from S to D via an out-of-band channel and are then displayed on P.

Note that Theorems 5 and 6 assume that the human H has no initial knowledge. This may appear rather strong as, in reality, humans know many things including PINs and passwords. The following theorem states the simple topological condition for which HISPs providing secure channels exist under the assumption that there are secret terms in the initial knowledge of H and S. The only condition is that there exists a communication path.

Theorem 7. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology. If H and S share two secret fresh constants and there is a path from H to S (from S to H) in τ then there exists a protocol providing a secure channel from H to S (from S to H).*

To see why this theorem is true, suppose that H and S have the term $\langle x, y\rangle$ in their initial knowledge, where x and y are fresh constants, not known to the dishonest agent P. Then H sends x to securely communicate y to S. Such protocols are of marginal interest in practice. In particular, x is a term that can be used only once, and which a human would typically read off of a code sheet. But code sheets are modeled in HISPs as a supporting technology D and reading the code sheet is represented by the edge (D, H).

V. Related Work

Security ceremonies were informally introduced by Ellison [10], [27] as a generalization of security protocols. They have given rise to several formal models that we discuss below. Our model is both more abstract and more precise than Ellison's description of security ceremonies.

Bella and Coles-Kemp extend security ceremonies with socio-technical elements such as a human agent's belief system and cultural values [2], [3]. They propose modeling security ceremonies using five layers: (1) the security of the protocol executed by the computers of the communicating partners; (2) the inter-process communication of the operating system; (3) human-computer interaction; (4) the user's state of mind; and (5) the influence of society on individuals. In [3], they formalize layer (3) and give a case study verifying a user's confidence in the privacy assurance offered by a service provider in an example ceremony. In contrast to Bella and Coles-Kemp's work, we prove general results about secure communication scenarios that involve a human and his compromised computer.

Meadows and Pavlovic propose a logic of networks involving humans, devices, and computers. They analyze various authentication protocols [21] with respect to claimed security guarantees, but they do not provide a formal attacker model. Their formalism is comprehensive, but complex. In subsequent work, they extend their logic to a "logic of moves" and use it to analyze physical airport security procedures [18]. Similarly to Meadows and Pavlovic, we provide a graphical model for the communication topologies of security ceremonies. However, our abstraction is simpler while supporting the modeling of the communication topologies of security ceremonies in arbitrary detail. The level of abstraction we use is both intuitive to understand and straightforward to verify with existing protocol verification tools. Moreover, we provide a comprehensive formal attacker model for the verification of security properties of protocols involving humans, devices, and computers.

Carlos et al. sketch a method to formalize human knowledge distribution in security ceremonies [6]. In subsequent work [7], they consider an adversary that is weaker than the standard Dolev-Yao adversary in order to verify a Bluetooth pairing ceremony under realistic conditions. Their results are, however, specific to Bluetooth pairing ceremonies.

Other related research areas address the *secure platform problem* [23], the *problem of untrusted terminals* [4], and *trusted paths* [12], [29]. The first two deal with the problem of ensuring that the user's computing platform faithfully executes a security protocol and does not leak confidential information to any unintended third party. The third is the problem of providing secure channels from an input device to a trusted application and onward to an output device and focuses on implementation details at the system level.

Regarding our formalization of insecure, authentic, and confidential channels, Mödersheim and Viganò provide a security protocol model [20] based on abstract channels as assumptions and goals. Their *ideal channel model* is related to our channel rules in that it provides an abstract notation for sending messages via authentic and confidential channels. Whereas Mödersheim and Viganò implement their abstract channels using asymmetric cryptography, our channel rules directly specify the adversary's interaction with the abstract channels.

VI. Conclusions

We have introduced a formal model for security protocols operating in an environment with humans, computers, and devices as actors. The salient feature of our model is the communication topology, which is a labeled graph whose vertices and edges represent the actors and their communication channels. The vertex labeling represents the assumptions made about the actors' initial knowledge, computational capabilities, and honesty. The edge labeling assigns channel assumptions (such as being confidential, authentic, or insecure) to communication links. These assumptions determine whether secure communication is possible between two nodes in the topology. We have demonstrated the usefulness of our model by completely characterizing the necessary and sufficient conditions for the existence of HISPs, which is the class of security protocols where a human securely communicates with a remote server while using a compromised computer platform. Our model is supported by Tamarin [19], a security protocol verification tool and our examples show applications of our modeling approach and its tool support.

Our characterization of HISPs answers the question of which secure or insecure communication channels must be available to establish a secure communication channel between a human and a remote server. There are several related questions that could be posed and our work paves the way for finding their answers. For instance, we could distinguish between different types of trusted devices, in terms of cost, sophistication (paper versus smart cards), or levels of trust that depend on whether secrets must be stored on the device. We could also consider a wider variety of channel properties, for example, what if the channel between human and trusted device is authentic, but not confidential and the adversary cannot replay messages on the channel? Furthermore, there are different communication topologies that would benefit from a similar analysis. An example is the problem of distributing cryptographic keys and firmware updates to the large variety of smart items that form the "Internet of Things".

References

[1] D. Basin, S. Radomirović, and M. Schläpfer. A complete characterization of secure human-server communication (full version), April 2015. http://www.infsec.ethz.ch/research/projects/hisp.html.

[2] G. Bella and L. Coles-Kemp. Seeing the full picture: the case for extending security ceremony analysis. In *Proceedings of 9th Australian Information Security Management Conference*, pages 49–55, 2011.

[3] G. Bella and L. Coles-Kemp. Layered analysis of security ceremonies. In D. Gritzalis, S. Furnell, and M. Theoharidou, editors, *Information Security and Privacy Research*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 273–286. Springer, 2012.

[4] I. Berta. *Mitigating the attacks of malicious terminals*. PhD thesis, Budapest University of Technology and Economics, 2005.

[5] C. Caleiro, L. Viganò, and D. A. Basin. On the semantics of Alice & Bob specifications of security protocols. *Theor. Comput. Sci.*, 367(1-2):88–122, 2006.

[6] M. C. Carlos, J. E. Martina, G. Price, and R. F. Custódio. A proposed framework for analysing security ceremonies. In P. Samarati, W. Lou, and J. Zhou, editors, *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography*, pages 440–445. SciTePress, 2012.

[7] M. C. Carlos, J. E. Martina, G. Price, and R. F. Custódio. An updated threat model for security ceremonies. In *28th Symposium on Applied Computing*, pages 1836–1843. ACM, 2013.

[8] D. Chaum. SureVote: Technical overview. In *Proceedings of the workshop on trustworthy elections (WOTE'01)*, 2001.

[9] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[10] C. M. Ellison. Ceremony design and analysis. *IACR Cryptology ePrint Archive*, 2007:399, 2007.

[11] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner. A survey of mobile malware in the wild. In X. Jiang, A. Bhattacharya, P. Dasgupta, and W. Enck, editors, *SPSM'11, Proceedings of the 1st ACM Workshop Security and Privacy in Smartphones and Mobile Devices*, pages 3–14. ACM, 2011.

[12] A. Filyanov, J. M. McCuney, A.-R. Sadeghiz, and M. Winandy. Uni-directional trusted path: Transaction confirmation on just one device. In *IEEE/IFIP 41st Intl. Conf. on Dependable Systems & Networks (DSN)*, pages 1–12. IEEE, 2011.

[13] S. Gajek. A universally composable framework for the analysis of browser-based security protocols. In J. Baek, F. Bao, K. Chen, and X. Lai, editors, *Provable Security*, volume 5324 of *LNCS*, pages 283–297. Springer, 2008.

[14] T. Groß, B. Pfitzmann, and A.-R. Sadeghi. Browser model for security analysis of browser-based protocols. In S. Vimercati, P. Syverson, and D. Gollmann, editors, *Computer Security – ESORICS 2005*, volume 3679 of *LNCS*, pages 489–508. Springer, 2005.

[15] A. Herzberg and R. Margulies. Forcing Johnny to login safely. In V. Atluri and C. Diaz, editors, *Computer Security – ESORICS 2011*, volume 6879 of *LNCS*, pages 452–471. Springer, 2011.

[16] A. Hiltgen, T. Kramp, and T. Weigold. Secure internet banking authentication. *Security & Privacy, IEEE*, 4(2):21–29, 2006.

[17] U. Maurer and P. Schmid. A calculus for secure channel establishment in open networks. In D. Gollmann, editor, *Computer Security – ESORICS 94*, volume 875, pages 173–192. Springer, 1994.

[18] C. Meadows and D. Pavlovic. Formalizing physical security procedures. In A. Jøsang, P. Samarati, and M. Petrocchi, editors, *Security and Trust Management*, volume 7783 of *LNCS*, pages 193–208. Springer, 2013.

[19] S. Meier, B. Schmidt, C. Cremers, and D. Basin. The TAMARIN prover for the symbolic analysis of security protocols. In N. Sharygina and H. Veith, editors, *25th International Conference on Computer Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pages 696–701. Springer, July 2013.

[20] S. Mödersheim and L. Viganò. Secure pseudonymous channels. In M. Backes and P. Ning, editors, *Computer Security – ESORICS 2009*, volume 5789 of *LNCS*, pages 337–354. Springer, 2009.

[21] D. Pavlovic and C. Meadows. Actor-network procedures. In R. Ramanujam and S. Ramaswamy, editors, *Distributed Computing and Internet Technology*, volume 7154 of *LNCS*, pages 7–26. Springer, 2012.

[22] M. Polychronakis, P. Mavrommatis, and N. Provos. Ghost turns zombie: Exploring the life cycle of web-based malware. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, LEET'08, page 8. USENIX Association, 2008.

[23] R. Rivest. *Perspective on Electronic voting*, volume 2339 of *LNCS*, chapter "The Business of Electronic Voting (Panel)", pages 243–268. Springer, 2001.

[24] M. Schläpfer and M. Volkamer. The secure platform problem: Taxonomy and analysis of existing proposals to address this problem. In *6th International Conference on Theory and Practice of Electronic Governance, ICEGOV '12*, pages 410–418. ACM, 2012.

[25] B. Schmidt, S. Meier, C. Cremers, and D. Basin. Automated analysis of Diffie–Hellman protocols and advanced security properties. In *25th IEEE Computer Security Foundations Symposium, CSF 2012*, pages 78–94. IEEE, 2012.

[26] B. Schmidt, P. Schaller, and D. Basin. Impossibility results for secret establishment. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010*, pages 261–273. IEEE Computer Society, 2010.

[27] UPnP Security Working Group. UPnP^{TM} security ceremonies, October 2003.

[28] T. Weigold, T. Kramp, R. Hermann, F. Höring, P. Buhler, and M. Baentsch. The Zurich Trusted Information Channel–an efficient defence against man-in-the-middle and malicious software attacks. In *Trusted Computing-Challenges and Applications*, volume 4968 of *LNCS*, pages 75–91. Springer, 2008.

[29] Z. Zhou, V. D. Gligor, J. Newsome, and J. M. McCune. Building verifiable trusted path on commodity x86 computers. In *Security and Privacy (S&P), 2012 IEEE Symposium on*, pages 616–630. IEEE, 2012.

Appendix

The first two sections give details on the Tamarin model [25] and our extensions. The remaining two sections give statements of lemmas. Proof details are given in the full version [1].

A. Term Algebra and Adversary Model

a) Term Algebra: The term algebra is order-sorted with the sort msg and its two incomparable subsorts $fresh$ and pub. There are two countably infinite sets $\mathcal{C}_{fresh}$ and $\mathcal{C}_{pub}$ of fresh and public constants, respectively, and we denote their union by $\mathcal{C}$. Let $S := \{fresh, pub, msg\}$. For each sort $s \in S$, there is a countably infinite set $\mathcal{V}_s$ of variables. We write $x{:}s$ to denote that $x \in \mathcal{V}_s$ and we let $\mathcal{V} := \bigcup_{s \in S} \mathcal{V}_s$.

A signature Σ is a set of function symbols, where each function symbol is associated with an arity. The subset of n-ary function symbols is denoted by Σ^n and we set $\Sigma^0 = \mathcal{C}_{fresh} \cup \mathcal{C}_{pub}$. Messages are elements of the term algebra $\mathcal{T} = T(\Sigma, \mathcal{V})$, and ground terms are elements of $\mathcal{M} = T(\Sigma, \emptyset)$.

In this paper we assume that $\Sigma = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3$, where

$$\begin{aligned}
\Sigma^1 &= \{\pi_1(\_),\ \pi_2(\_),\ \mathrm{h}(\_),\ \mathrm{pk}(\_)\} \\
\Sigma^2 &= \{\langle \_,\_ \rangle,\ \mathrm{senc}(\_,\_),\ \mathrm{sdec}(\_,\_), \\
&\qquad \mathrm{aenc}(\_,\_),\ \mathrm{adec}(\_,\_),\ \mathrm{sign}(\_,\_)\}, \\
\Sigma^3 &= \{\mathrm{verify}(\_,\_,\_)\}.
\end{aligned}$$

For $i > 0$, all functions in Σ^i are of sort $msg \times \cdots \times msg \to msg$. The function $\langle \_,\_ \rangle$ represents the pairing of terms, and π_1 and π_2 are the first and second projections, respectively. The functions $\mathrm{senc}(\_,\_)$ and $\mathrm{aenc}(\_,\_)$ represent symmetric and asymmetric encryption and $\mathrm{sdec}(\_,\_)$ and $\mathrm{adec}(\_,\_)$ represent symmetric and asymmetric decryption, respectively. The functions $\mathrm{sign}(\_,\_)$ and $\mathrm{verify}(\_,\_,\_)$ represent signing and verification of signatures. $\mathrm{h}(\_)$ represents a hash function and $\mathrm{pk}(\_)$ corresponds to the public key for a given secret key. For $a, b \in \mathcal{T}$, $true \in \mathcal{C}_{pub}$, we let $\mathcal{E}$ be the following set of equations over Σ:

$$\begin{aligned}
\{\ &\pi_1(\langle a,b \rangle) = a,\ \pi_2(\langle a,b \rangle) = b, \\
&\mathrm{sdec}(\mathrm{senc}(a,b),b) = a,\ \mathrm{adec}(\mathrm{aenc}(a,\mathrm{pk}(b)),b) = a, \\
&\mathrm{verify}(\mathrm{sign}(a,b),a,\mathrm{pk}(b)) = true\ \}.
\end{aligned}$$

The equational theory $Eq(\Sigma, \mathcal{E})$ is the smallest congruence containing all instances of the equations of $\mathcal{E}$ over Σ.

A position p is a (possibly empty) sequence of positive natural numbers. The subterm $t_{|p}$ of t at position p is inductively defined by t if p is empty and by $(t_i)_{|p'}$ if $p = [i] \cdot p'$ and $t = f(t_1, \ldots, t_n)$ for $f \in \Sigma^n$ and $1 \leq i \leq n$. The set of all subterms of t is denoted by $St(t)$. The set of variables of t is denoted by $vars(t) := St(t) \cap \mathcal{V}$.

b) Adversary model: The network is controlled by a Dolev-Yao adversary [9]. The adversary chooses whether to deliver each message. He eavesdrops on, injects, and modifies messages on channels. However, he can neither eavesdrop on confidential (or secure) channels nor inject or modify messages on authentic (or secure) channels. The message deduction rules in $\mathcal{MD}$ represent his capability to receive, construct, and send messages in a protocol execution:

$$\begin{aligned}
\mathcal{MD} \;:=\; & \{[\mathsf{Out}(x)] \xrightarrow{!\mathsf{K}(x)} [!\mathsf{K}(x)], && (22)\\
& [!\mathsf{K}(x)] \xrightarrow{!\mathsf{K}(x)} [\mathsf{In}(x)], && (23)\\
& [\,] \xrightarrow{!\mathsf{K}(x:pub)} [!\mathsf{K}(x:pub)], && (24)\\
& [\mathsf{Fr}(x)] \xrightarrow{!\mathsf{K}(x)} [!\mathsf{K}(x)]\} && (25)\\
\cup \; & \{[!\mathsf{K}(x_1), \ldots, !\mathsf{K}(x_k)] \xrightarrow{!\mathsf{K}(f(x_1,\ldots,x_n))} && (26)\\
& \quad [!\mathsf{K}(f(x_1,\ldots,x_n))] \mid f \in \Sigma^n \wedge n > 0 \;\}.
\end{aligned}$$

The !K fact appearing in all rules of $\mathcal{MD}$ is used to store and observe the adversary's knowledge in a trace and plays a role in specifying secrecy properties.[2] Rule (22) allows the adversary to learn all terms that are produced with Out facts and rule (23) allows the adversary to input any term in his knowledge into an In fact. The Rules (24) and (25) represent the adversary's capabilities to learn public and freshly generated constants, respectively. The set of Rules (26) allow the adversary to apply any function in Σ^n, for $n > 0$, to known messages.

B. Extended Model Details

We provide here additional details on our model extensions.

The following definition summarizes all facts used in the model.

$$\begin{aligned}
\Sigma_{Fact} :=& \Sigma^1_{Fact} \cup \Sigma^2_{Fact} \cup \Sigma^3_{Fact}, \text{ where}\\
\Sigma^1_{Fact} :=& \{\mathsf{Fr}, \mathsf{Out}, \mathsf{In}, !\mathsf{K}, \mathsf{Honest}, \mathsf{Dishonest}, \mathsf{Trust}\},\\
\Sigma^2_{Fact} :=& \{!\mathsf{Auth}, !\mathsf{Conf}, \mathsf{Fresh}, \mathsf{Comm}, \mathsf{Learn}\},\\
\Sigma^3_{Fact} :=& \{\mathsf{Snd_I}, \mathsf{Rcv_I}, \mathsf{Snd_A}, \mathsf{Rcv_A}, \mathsf{Snd_C}, \mathsf{Rcv_C}, \mathsf{Snd_S}, \mathsf{Rcv_S}\}\\
& \cup \{!\mathsf{Sec}, \mathsf{Secret}, \mathsf{Authentic}, \mathsf{AgSt}\}.
\end{aligned}$$

The set of all facts $\mathcal{F}$ is therefore

$$\mathcal{F} := \{f(t_1, \ldots, t_n) \mid f \in \Sigma^n_{Facts} \wedge t_1, \ldots, t_n \in \mathcal{T}\}.$$

We use the action $\mathsf{Honest}(A)$ to label an agent A honest and $\mathsf{Dishonest}(A)$ to label the agent dishonest in a trace. Once an agent is labeled honest, it cannot become dishonest or vice-versa. In particular, if an agent A is labeled honest, then a rule that contains the action $\mathsf{Dishonest}(A)$ cannot be applied. This is enforced in Tamarin with an axiom. Trust is used to label agents that are assumed to be honest for the purpose of security properties, see Definition 7. These are agents whose roles are marked honest in the communication topology.

We distinguish between model and protocol specification rules, denoted by $\mathcal{R}_{Model}$ and $\mathcal{R}_{Spec}$ respectively. The former are the fixed set of rules

$$\mathcal{R}_{Model} := \{[\,] \to [\mathsf{Fr}(x{:}fresh)]\} \cup \mathcal{MD} \cup \mathcal{CH} \cup \mathcal{AG}$$

introduced in Section III-B and Appendix A. The latter specify the security protocol. Recall that Rule (1) is the only rule producing fresh constants and thereby creating Fr facts. Fresh constants can be obtained (generated) by honest agents using Rule (2). Dishonest agents obtain fresh constants from the adversary using Rule (5). The adversary can generate fresh constants using Rule (25).

A protocol defines a *setup* and the behavior of a set of *roles*. The corresponding protocol specification $\mathcal{R}_{Spec}$ consists of a finite number of setup rules and protocol rules. Setup rules are used to initialize the protocol, i.e., to generate the initial knowledge and to distribute it to the corresponding protocol agents by generating the initial AgSt facts for all roles. Formally, a setup rule $l \xrightarrow{a} r$ is a rule where:

S1 Only Fr facts occur in l.
S2 The actions Learn, Comm, Secret, Authentic, and Trust do not occur in a.

A role consists of a set of protocol rules, specifying the sending and receiving of messages, branching and looping conditions, and the generation of fresh constants. In what follows, we only allow protocols where after the setup phase all information is exchanged using the channels defined in our channel abstraction model above. That is, information may not flow from one agent to another in any way other than by one of the channels defined in $\mathcal{CH}$. A protocol rule $l \xrightarrow{a} r$ is a rule such that the following 5 conditions are satisfied.

P1 The facts in l, a, and r do not contain elements of $\mathcal{C}_{fresh}$ as subterms.
P2 Only $\mathsf{Rcv_I}, \mathsf{Rcv_A}, \mathsf{Rcv_C}, \mathsf{Rcv_S}$, and Fresh facts and exactly one AgSt fact occur in l.
P3 Only $\mathsf{Snd_I}, \mathsf{Snd_A}, \mathsf{Snd_C}, \mathsf{Snd_S}$, and AgSt facts occur in r.
P4 If $\mathsf{AgSt}(A, step, kn)$ occurs in l, then:
(a) Every $\mathsf{Rcv_I}$, $\mathsf{Rcv_A}$, $\mathsf{Rcv_C}$, $\mathsf{Rcv_S}$, and Fresh fact is of the form $\mathsf{Rcv_I}(B, A, x)$, $\mathsf{Rcv_A}(B, A, x)$, $\mathsf{Rcv_C}(B, A, x)$, $\mathsf{Rcv_S}(B, A, x)$, and $\mathsf{Fresh}(A, x)$, where $B, x \in \mathcal{T}$.
(b) Every Learn, Comm, Secret, Authentic, $\mathsf{Snd_I}$, $\mathsf{Snd_A}$, $\mathsf{Snd_C}$, and $\mathsf{Snd_S}$ fact is of the form $\mathsf{Learn}(A, x)$, $\mathsf{Comm}(A, x)$, $\mathsf{Secret}(A, B, x)$, $\mathsf{Authentic}(B, A, x)$, $\mathsf{Snd_I}(A, B, x)$, $\mathsf{Snd_A}(A, B, x)$, $\mathsf{Snd_C}(A, B, x)$, and $\mathsf{Snd_S}(A, B, x)$, where $B \in \mathcal{C}_{pub}$, $x \in \mathcal{T}$ and x is derivable from terms in $\mathcal{C}_{pub}$, terms in Fresh and $\mathsf{Rcv_I}$, $\mathsf{Rcv_A}$, $\mathsf{Rcv_C}$, and $\mathsf{Rcv_S}$ facts occurring in l, and terms in kn.
(c) Every AgSt fact in r is $\mathsf{AgSt}(A, step', kn')$, where $step' \in \mathcal{C}_{pub}$ and kn' is derivable from terms in $\mathcal{C}_{pub}$, terms in Fresh and $\mathsf{Rcv_I}$, $\mathsf{Rcv_A}$, $\mathsf{Rcv_C}$, and $\mathsf{Rcv_S}$ facts occurring in l, and terms in kn.
P5 $vars(r) \subseteq vars(l) \cup \mathcal{V}_{pub}$.

Remark. *A protocol rule that contains a receive fact in its premise and a send fact in its conclusion models the reception and sending of messages as an* atomic *protocol execution step. If the agent executing the protocol step is dishonest, then the adversary may not be able to influence the message to be sent. To model the general situation where reception and the subsequent sending of messages are not atomic, two separate rules must be specified, one for the reception of messages and*

[2]For efficiency reasons, Tamarin distinguishes between !KU and !KD facts. For simplicity, we refer to both of these as !K facts.

a corresponding update of the receiver's state, and a second one to specify the sending of messages. The adversary may then reveal and modify a dishonest agent's state after the dishonest agent receives a message and before the agent sends the subsequent message.

C. Impossibility Results

In this appendix, we state impossibility lemmas for our main characterization results.

Lemma 3. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology where $K_H = \emptyset$ and no edge between H and D exists. Then there exists no protocol for τ that provides a confidential channel and there exists no protocol for τ that provides an authentic channel from H to S or vice-versa.*

The key idea for the proofs of Lemma 3 and the following lemmas is that every trace establishing a confidential or authentic channel that involves actions of D, can be transformed into a valid trace with the same properties but not involving D. Since the channels between H and S are insecure, by Lemmas 1 and 2 neither confidential nor authentic channels can be established between H and S.

Lemma 4. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology with $K_H = \emptyset$, $(H, D) \in E$, $(D, H) \notin E$, and $(D, S) \notin E^+$. Then there exists no protocol for τ that (1) provides a confidential channel or (2) provides an authentic channel from H to S.*

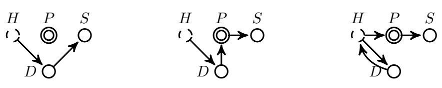
Lemma 5. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology with $K_H = \emptyset$, $(H, D) \in E$, $(S, H) \in E^+$, and $(D, H) \notin E^+$. Then there exists no protocol for τ that (1) provides a confidential channel or (2) provides an authentic channel from S to H.*

D. Possibility Results

The following lemmas assert the existence of HISPs that provide secure channels between H and S for the topologies not covered by the impossibility results above. Our proofs [1] embody protocols that we have verified using Tamarin.

Lemma 6. *Let $\tau = (V, E, \eta, \mu)$ be any HISP topology with $(H, D) \in E$ and $(D, S) \in E^+$. Then there exists a protocol for τ that provides an originating secure channel from H to S, even if $K_H = \emptyset$.*

The following graphs consist of an acyclic path from D to S and an additional edge $(H, D) \in E$.

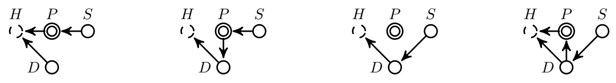


(a) (b) (c)

Lemma 7 states that for HISP topologies containing an edge from D to H, there is a protocol providing a secure channel from S to H, if there is a path from S to H.

Lemma 7. *Let (V, E, η, μ) be a HISP topology with $(S, H) \in E^+$. If $(D, H) \in E$ then there exists a protocol that provides a secure channel from S to H, even if $K_H = \emptyset$.*

The following are all acyclic paths from S to H together with an additional edge $(D, H) \in E$.

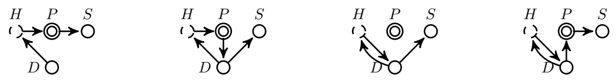


(a) (b) (c) (d)

Lemma 8 states that for HISP topologies containing an edge from D to H, there is a protocol providing a secure channel from H to S, if there is a path from H to S.

Lemma 8. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology with $(H, S) \in E^+$. If $(D, H) \in E$ then there exists a protocol for τ that provides a secure channel from H to S, even if $K_H = \emptyset$.*

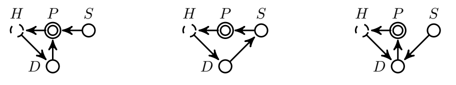
The following are all acyclic paths from H to S together with an additional edge $(D, H) \in E$.



(a) (b) (c) (d)

Lemma 9. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology with $(S, H) \in E^+$ and $(D, H) \notin E$. If $(H, D) \in E$ and $(D, H) \in E^+$, then there exists a protocol for τ that provides an originating authentic channel from S to H, even if $K_H = \emptyset$.*

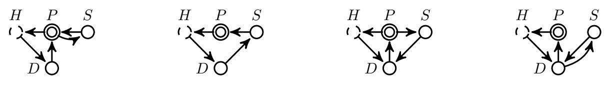
The minimal graphs satisfying the lemma's hypothesis are obtained as follows. There are two acyclic paths from S to H with $(D, H) \notin E$. One satisfies $(D, H) \in E^+$ and leads to case (c). The other leads to cases (a) and (b), since there are two acyclic paths form D to H with $(D, H) \notin E$.



(a) (b) (c)

Lemma 10. *Let $\tau = (V, E, \eta, \mu)$ be a HISP topology with $K_H = \emptyset$, $(S, H) \in E^+$, $(D, H) \notin E$, $(H, D) \in E$, and $(D, H) \in E^+$. Then there exists a protocol for τ that provides a secure channel from S to H if and only if $(H, S) \in E^+$.*

The minimal graphs satisfying the lemma's hypothesis are obtained from the graphs of Lemma 9 and the additional condition $(H, S) \in E^+$.



(a) (b) (c) (d)