Modeling human errors in security protocols

Basin, David; Radomirovic, Sasa; Schmid, Lara

Published in:
IEEE 29th Computer Security Foundations Symposium CSF 2016

DOI:
10.1109/CSF.2016.30

Publication date:
2016

Document Version
Peer reviewed version

Modeling Human Errors in Security Protocols

(Full Version)

David Basin Saša Radomirović Lara Schmid
Institute of Information Security, Department of Computer Science, ETH Zürich
Email: {david.basin, sasa.radomirovic, schmidla}@inf.ethz.ch

***Abstract*—Many security protocols involve humans, not machines, as endpoints. The differences are critical: humans are not only computationally weaker than machines, they are naive, careless, and gullible. In this paper, we provide a model for formalizing and reasoning about these inherent human limitations and their consequences. Specifically, we formalize models of fallible humans in security protocols as multiset rewrite theories. We show how the Tamarin tool can then be used to automatically analyze security protocols involving human errors. We provide case studies of authentication protocols that show how different protocol constructions and features differ in their effectiveness with respect to different kinds of fallible humans. This provides a starting point for a fine-grained classification of security protocols from a usable-security perspective.**

I. Introduction

Humans use and interact with security protocols in many contexts, for example during e-banking or to cast their vote in electronic elections. In contrast to protocols where only machines communicate with each other and precisely follow the protocol specification, new opportunities for attacks arise when humans are involved. It is possible that users do not understand what they should and should not do and even knowledgeable users may neglect to perform some protocol steps due to carelessness.

Attackers, of course, are well aware of human fallibility and exploit this in their attacks. Humans are often targeted because it is easier to get information or access to a system by social engineering rather than by directly attacking machines or breaking the underlying cryptography. For example, many people are fooled by phishing attacks into simply giving away their secret credentials. Moreover, humans are bad at identifying phishing websites even when they are specifically instructed to do so in controlled lab environments [4], [12]. Hence even when security is their primary concern and they are attentive, humans are incapable of performing basic security checks. This problem is exacerbated in everyday situations where security is a secondary concern. Despite the severity of this problem, human weaknesses have received little attention in security protocol analysis. Since there are situations where human interaction in protocols is unavoidable [10], we must be able to analyze this interaction.

In this work, we propose a formal model of communication protocols that includes humans and their fallibilities. We define a *human error* as any deviation of a human from the protocol specification. We identify two natural approaches to defining fallible humans that may deviate from the protocol specification to different extents. We then single out one of these approaches where we consider fallible humans that can take arbitrary steps instead of following a specification. This models non-expert humans in everyday protocols.

To support automated reasoning, we build on an existing formalization of security protocols as multiset rewriting theories and extend it with fallible, non-expert humans. The resulting formalism is supported by the automatic verification tool Tamarin [20], [25] and enables the unbounded verification of security protocols.

We validate our formal model in two case studies. First, we examine an authentication protocol and a proposed improved version thereof. Second, we analyze different phone-based authentication protocols and compare their security guarantees with respect to different kinds of fallible humans. All these protocols have the goal of authenticating a human agent to a remote server and all of them succeed with infallible humans. However Tamarin finds numerous and varied attacks on these protocols arising from different kinds of human errors.

Contributions: We present the first formal model of human errors in security protocols. This model makes precise the notion that humans are fallible and may not behave as expected. It thereby bridges the gap between formal models of security protocols that fail to consider human errors and the empirical analyses of protocols that demonstrate how these errors can lead to attacks.

We identify different approaches to defining fallible humans and show how one of them can be formalized and integrated into an existing formalization of security protocols. The resulting model is supported by Tamarin and allows for the unbounded verification of security protocols involving fallible humans. In addition to a distinguished human H for whom security properties are analyzed, we allow for an arbitrary number of fallible humans in the network. This allows errors made by other humans to affect the security guarantees that hold for H.

We present two applications of our model to automatically finding attacks arising from human errors. First, we analyze an authentication protocol in detail and compare it with a modified version that uses a heuristic to avoid human errors. Second, we compare existing phone-based authentication protocols that use different methods to authenticate a human to a remote server. We examine different authentication properties and consider humans with different skills. The model helps us identify which methods provide effective protection against which kinds of fallible humans.

Our applications show that our model provides a formal

basis to understand different security practices and their effects. For instance we identify what information is useful to a human to achieve a given protocol's security goals. Moreover, the model allows us to make fine-grained distinctions between protocols with respect to their resilience to different kinds of human errors. This adds a new dimension to the framework proposed by *Bonneau et al.* [8], where all but one of the protocols have previously been compared.

Organization: We present in Section II different approaches to defining fallible humans. We give an overview of an existing formal protocol model in Section III and enhance it in Section IV to account for fallible humans. Afterward, we present two applications of our model. In Section V, we analyze one protocol in detail and in Section VI we compare different protocols for authenticating a human to a server. We discuss related work in Section VII and draw conclusions in Section VIII.

II. Approaches to Defining Fallible Humans

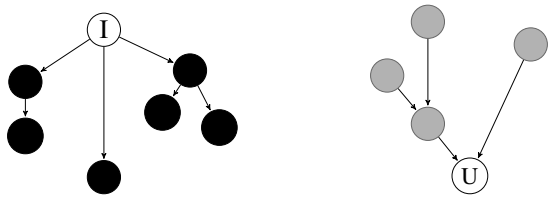
We start with a simple model of human capabilities. We assume that humans can perform simple computations but not complex operations. That is, they are able to send and receive messages on specified channels, concatenate (i.e., pair) messages, and split concatenated messages. In our model, humans cannot, unaided, encrypt or decrypt messages; they require computers for this.

A protocol defines the behaviors of agents (computers or humans) by specifying their *role*. We model human agents by keeping track of their knowledge and allowing other agents and the adversary to query and update the human's knowledge. This models that a human can store information from any source and communicate his knowledge to others. We do not limit the number of terms that humans can remember. We define human error considering only a human's behavior, but neither his intention nor the reasons that lead to his errors.

Definition 1. *A* human error *in a protocol execution is any deviation of a human from his or her role specification. Such a human is said to be* fallible*. A human that does not deviate from his role specification is said to be* infallible*.*

A fallible human gives rise to more system behaviors than an infallible human because, in addition to following the protocol specification, the fallible human can deviate from it. Human agents can therefore be partially ordered by the sets of system behaviors they allow. This leads us to propose two natural approaches to analyzing the security of protocols in the presence of human errors: The *skilled human* approach, where we consider increasingly fallible humans by adding possibilities for human error and the *rule-based human* approach where we remove possibilities for human error.

The two approaches are inspired by the human performance levels that psychologists differentiate between in the GEMS model [23]. At the skill-based level, a human behaves according to familiar detailed patterns that have led her to the desired or a similar goal before, but she can err. At the rule-based performance level, a human acts according to general rules



(a) I = Infallible Human (b) U = Untrained Human

Fig. 1: Skilled and Rule-Based Human Approaches: Black and gray nodes represent skilled and rule-based humans, respectively. An arrow denotes that the human at the tail node induces only a subset of the system behaviors that the human at the arrow's tip induces.

of the form "in situation X perform action Y". In contrast to psychological models, we assume that a rule-based human will never apply a rule incorrectly; he will always follow exactly the rules he knows. We discuss the two approaches next. However, we focus in this work on the rule-based approach.

A. Skilled Humans

The skilled human approach starts from an infallible human agent, depicted by the node *I* at the top of the hierarchy in Figure 1a. In this approach, human errors are modeled by weakening the infallible human agent to a *skilled human* agent who can make a fixed number of mistakes. Thus while the infallible human agent represents an expert human who never errs, the skilled human agent represents a human that knows the protocol's steps, but can make a slip.

An example of a skilled human is an expert user that knows what she must do, but skips a verification step of the protocol due to inattentiveness. In Figure 1a, each black node in the hierarchy represents a skilled human agent. Depending on the specified mistakes that the human can make, more or fewer system behaviors are possible. This approach allows us to examine which specific errors lead to attacks on protocols.

B. Rule-Based Humans

The rule-based human approach starts from an *untrained human* agent depicted by node *U* at the bottom of the hierarchy in Figure 1b. The untrained human does not know the protocol specification and may blindly follow any adversarial instruction he is given. That is, the untrained human agent can perform any action permitted by the execution model. In this approach, *rule-based human* agents are defined by strengthening the untrained human agent with a set of rules that he must follow. This models a human that does not know the protocol specification but adheres to some basic guidelines.

An example of this is a human who knows he must only type his password into a trusted device, even if he does not know the detailed steps of the protocol he participates in. Rule-based human agents are shown as gray nodes in Figure 1b. While the untrained human can behave arbitrarily, the rules of a rule-based human restrict his possible behaviors and thus the system behaviors. We will fix a set of guidelines in Section IV-C, when we formally define rule-based humans.

III. Modeling Security Protocols

In this section, we summarize an existing security protocol model that we extend, in the next section, to account for fallible humans. The model presented here is based on the Tamarin model [20], [25] with some minor extensions for representing channels with security properties [5].

A. Preliminaries

The term algebra of *messages* is given by $\mathcal{T}_\Sigma(X)$, where Σ is a signature and X a disjoint, countably infinite set of variables. The ground terms are $\mathcal{T}_\Sigma$. $F_{sym} \subset \Sigma$ denotes a finite set of function symbols that always contains the functions $\mathrm{pair}(\_,\_)$, also denoted by $\langle \_,\_ \rangle$, for pairing, $\pi_1(\_)$ and $\pi_2(\_)$ for the first and second projection of a pair of terms, and $\mathrm{h}(\_)$ for hashing terms. Moreover, F_{sym} contains function symbols for symmetric and asymmetric encryption as well as creating and verifying digital signatures. For a message m and a key k, the functions $\mathrm{senc}(m,k)$ and $\mathrm{sdec}(m,k)$ denote symmetric encryption and decryption, $\mathrm{aenc}(m,k)$ and $\mathrm{adec}(m,k)$ denote asymmetric encryption and decryption, and $\mathrm{sign}(m,k)$ and $\mathrm{verify}(\mathit{sign},m,k)$ are used for signing messages and verifying signatures. The function $\mathrm{pk}(k)$ represents the public key corresponding to the private key k. The standard equational theory for these functions is given in [5, Appendix A].

Σ also contains the two countably infinite sets of fresh and public constants, denoted by $\mathcal{C}_{fresh}$ and $\mathcal{C}_{pub}$, respectively. Fresh constants model the generation of nonces, whereas public terms represent agent names and other publicly known values. The sets F_{sym}, $\mathcal{C}_{fresh}$ and $\mathcal{C}_{pub}$ are pairwise disjoint. We denote sequences with square brackets and use the operator $\cdot$ to concatenate sequences.

B. Protocol Specification

We use the extended Alice&Bob notation of [5] to specify security protocols. A simple protocol specification, for a fictitious protocol called *SimpleProtocol*, is shown in Figure 2. The protocol specifies two roles named S and R. Initially S knows R, while R knows S and the message m_2, as indicated with the keyword knows. In the first step, S generates a fresh message m_1, indicated with the keyword fresh, and sends it to R over an insecure channel, denoted by $\circ\!\!\rightarrow\!\!\circ$. Then R responds with the message m_2 over a secure channel, denoted by $\bullet\!\!\rightarrow\!\!\bullet$. This channel notation is taken from Maurer and Schmid's channel calculus [19]. In addition to the two channels shown in Figure 2, we shall use the expressions $A \bullet\!\!\rightarrow\!\!\circ\ B$ and $A \circ\!\!\rightarrow\!\!\bullet\ B$ to denote authentic and confidential channels from A to B, respectively. On an authentic channel, the adversary can read the communicated message but cannot modify the message or its sender. Conversely, the adversary cannot learn a message sent on a confidential channel. However, he can change the sender of a message sent on a confidential channel or send arbitrary messages from his own knowledge on it. A secure channel is a channel that is both authentic and confidential.

Role scripts are the projections of an extended Alice&Bob protocol specification to individual roles. They correspond to *strands* in the strand spaces model [27] and *processes* in the applied pi calculus [3]. Formally, a role script is a sequence of *events* $e \in \mathcal{T}_{\Sigma \cup \mathit{RoleActions}}(X)$, where $\mathit{RoleActions} = \{\mathtt{Send}, \mathtt{Receive}, \mathtt{Start}, \mathtt{Fresh}\}$ and the top-level function symbol of e is in $\mathit{RoleActions}$. Send and receive events are of the form $\mathtt{Send}(A,l,P,m)$ and $\mathtt{Receive}(A,l,P,m)$, where A is the role executing the event, $l \in \mathit{LinkProp} = \{\mathit{ins}, \mathit{auth}, \mathit{conf}, \mathit{sec}\}$ indicates the type of channel over which a message is sent, $P \in \mathcal{C}_{pub}$ is a role's name, and $m \in \mathcal{T}_\Sigma(X)$ is a message. The channel types ins, auth, conf, and sec denote insecure, authentic, confidential, and secure channels and correspond in the obvious manner to the channel symbols in the Alice&Bob notation. In a send event, the communication partner P is the intended recipient of the message m. In a receive event, the communication partner P is the apparent sender, as the adversary may have forged the message, and m is the expected message pattern. The fresh event $\mathtt{Fresh}(A,m)$ indicates that the role A generates a fresh message m and the start event $\mathtt{Start}(A,m)$ indicates the initial knowledge m of the role A. The start event is the first event of a role script and occurs only once.

0.		S :	$\mathrm{knows}(R)$
0.		R :	$\mathrm{knows}(S, m_2)$
1.	S	$\circ\!\!\rightarrow\!\!\circ$ R :	$\mathrm{fresh}(m_1).m_1$
2.	R	$\bullet\!\!\rightarrow\!\!\bullet$ S :	m_2

Fig. 2: Protocol *SimpleProtocol* in Alice&Bob notation.

$$[\mathtt{Start}(S,R), \mathtt{Fresh}(S,m_1), \mathtt{Send}(S, \mathit{ins}, R, m_1) \\ , \mathtt{Receive}(S, \mathit{sec}, R, m_2)]$$

Fig. 3: Role Script of role S for *SimpleProtocol*.

Figure 3 shows the role script of S for *SimpleProtocol*. The first event of the role script is the $\mathtt{Start}(S,R)$ event, where S is the name of the executing role and R is the role's initial knowledge. The second event $\mathtt{Fresh}(S,m_1)$ denotes that S generates the fresh value m_1. The third and fourth events denote that S sends m_1 to R over an insecure channel and receives m_2 from R over a secure channel, respectively.

C. Execution Model

We use Tamarin's execution model [20], [25], which is defined by a multiset term-rewriting system. A system state is a multiset of *facts*. There are two types of facts, *linear facts* and *persistent facts*. Linear facts model exhaustible resources and they can be added to and removed from the system state. Persistent facts model inexhaustible resources and can only be added to the system state. Persistent fact symbols are prefixed with an exclamation mark. The initial system state is the empty multiset. A trace tr is a finite sequence of multisets of actions a and is generated by the application of labeled state transition rules of the form $\mathit{prem} \xrightarrow{a} \mathit{conc}$. A state transition rule is applicable when the current state contains facts matching the premise prem. The rule's application removes the matching

$$[\quad] \xrightarrow{\texttt{Start}(S,R)} [\mathsf{AgSt}(S,0,R)] \quad \text{(S0)}$$

$$[\mathsf{AgSt}(S,0,R), \mathsf{Fr}(m_1)] \xrightarrow{\texttt{Fresh}(S,m_1)} [\mathsf{AgSt}(S,1,\langle R,m_1\rangle)] \quad \text{(S1)}$$

$$[\mathsf{AgSt}(S,1,\langle R,m_1\rangle)] \xrightarrow{\texttt{Send}(S,ins,R,m_1)} [\mathsf{AgSt}(S,2,\langle R,m_1\rangle), \mathsf{Out}_{ins}(\langle S,R,m_1\rangle)] \quad \text{(S2)}$$

$$[\mathsf{AgSt}(S,2,\langle R,m_1\rangle), \mathsf{In}_{sec}(\langle R,S,m_2\rangle)] \xrightarrow{\texttt{Receive}(S,sec,R,m_2)} [\mathsf{AgSt}(S,3,\langle R,m_1,m_2\rangle)] \quad \text{(S3)}$$

Fig. 4: Agent rules for agent S of protocol *SimpleProtocol*.

linear facts from the state, adds instantiations of the facts in the conclusion $conc$ to the state, and records the instantiations of actions in a in the trace. The set of all traces of a set of rules $\mathcal{R}$ is denoted by $TR(\mathcal{R})$.

We use the HISP model extension of Tamarin [5] to give a semantics to Alice&Bob protocols. A complete protocol model consists of the fresh rule, adversary rules, channel rules, and agent rules. The fresh rule

$$[\quad] \rightarrow [\mathsf{Fr}(x)] \quad \text{(F1)}$$

produces the fact $\mathsf{Fr}(x)$ where $x \in \mathcal{C}_{fresh}$. No two applications of this rule pick the same element $x \in \mathcal{C}_{fresh}$ and this is the only rule that can produce terms $x \in \mathcal{C}_{fresh}$. We use Tamarin's standard Dolev-Yao adversary [13] rules and the HISP channel rules to model the sending and receiving of messages over authentic, confidential, and secure channels. Agent rules specify the agents' state transitions and communication.

Given a role script of a role A, the corresponding agent rules are produced as follows. For every event e in the role script, we obtain a transition rule $prem \xrightarrow{a} conc$. The label of the rule contains the event, i.e., $e \in a$. The premise $prem$ contains an agent state fact $\mathsf{AgSt}(A, c, kn)$, which keeps track of the state of agent A, the protocol step c the agent is in, and the agent's knowledge kn. The conclusion $conc$ contains the subsequent agent state fact $\mathsf{AgSt}(A, c', kn')$. If e is a send event $\texttt{Send}(A, l, P, m)$, then the rule's $conc$ additionally contains an outgoing message fact $\mathsf{Out}_l(\langle A, P, m\rangle)$. If e is a receive event $\texttt{Receive}(A, l, P, m)$, then $prem$ contains an incoming message fact $\mathsf{In}_l(\langle P, A, m\rangle)$. If e is a fresh event $\texttt{Fresh}(A, x)$, then $prem$ contains a fresh fact $\mathsf{Fr}(x)$. Finally, if e is a start event $\texttt{Start}(A, i)$, then it is translated to a setup rule where $conc$ contains the initial agent state $\mathsf{AgSt}(A, 0, i)$.

Example 1. Consider the role script of S in *SimpleProtocol* shown in Figure 3. Recall that S first sends a fresh message m_1 over an insecure channel to R and then receives from R a message m_2 over a secure channel. The agent rules for the agent S are shown in Figure 4. The Setup Rule (S0) can be applied in every system state. Therefore, its premise is empty. The conclusion of the setup rule contains the Agent State 0 of S, with the initial knowledge R. The first premise of the rule (S1) is that S is in Agent State 0 where R is in its knowledge. The second premise $\mathsf{Fr}(m_1)$ denotes a fresh value m_1. Fr facts are produced by Rule (F1). In the conclusion, the knowledge in the agent state of S is updated with the fresh message m_1 and S proceeds to Agent State 1. For the send event (S2), the rule's premise consists of the current agent state. The effect of S sending a message to an insecure channel is that the fact Out_{ins} is now available in the network, expressed by the corresponding fact in the conclusion. The second fact in the conclusion expresses that agent S proceeds to the next agent state and keeps the same knowledge. Rule (S3) expresses that S receives the message m_2, from R, over a secure channel, which is denoted by the fact In_{sec} in the premise. In the conclusion, the knowledge in the agent state of S is updated with the received message m_2 and S proceeds to the next agent state.

D. Security Properties

We instrument protocol rules with additional actions in order to reason about the protocol's security properties. Following *Lowe* [17], we use the actions $\texttt{Running}(A, B, m)$ and $\texttt{Commit}(A, B, m)$ to define authentication properties. The actions are added to the rules corresponding to the protocol steps of an agent A, where A believes a given property to hold with respect to some partner B and message m.

We consider entity and message authentication [21]. *Entity authentication* states that one agent can be sure that another agent has the identity she claims and that she actually participated in the protocol: Whenever an agent B commits to a protocol execution with another agent A, then A was participating in the protocol recently. To capture recentness, we require that there is some event of A between the start event of B and B's claim. A formal definition of entity authentication is given in Definition 2 of Appendix A.

One way to authenticate humans is to ask them for something they know, like a password. If the correct password is received as a response, we conclude that the right person participated in the protocol. Another way is to check something the human possesses [21] such as a device D that he carries with him and to which he has exclusive access. If an agent B has the guarantee that a human H's device D recently participated in the protocol, then we say that *device authentication* of H to B holds. The assumption thereby is that D is only accessible to the human H that is being authenticated. See Definition 3 of Appendix A.

Message authentication holds when an agent B can be sure that a message was sent by another agent A. In contrast to entity authentication, we do not require the recentness of the sending event. A protocol provides message authentication of message m from an agent A to an agent B if whenever B believes to receive message m from A, then A previously sent m. See Definition 4 of Appendix A.

Finally, we define *secrecy* in standard manner: a message m is secret if the adversary does not learn it. See Definition 7 of Appendix A for secrecy's formal definition.

IV. Formally Modeling Fallible Humans

We now extend the above model to account for human errors. We first introduce the untrained-human rules and describe how infallible and untrained humans are modeled. Afterward, we introduce predicates to model rule-based humans.

A. Untrained-Human Rules

We construct agent rules for human roles in the same manner as for the other roles of a protocol shown in Section III. However, we also model that a human can make errors and deviate from the protocol specification. Therefore we introduce the *untrained human rules*. These rules model human knowledge as a database HK represented by key/value pairs that can be updated and queried by any agent, including the adversary. We refer to these pairs as tag/value pairs to avoid confusion with cryptographic keys. Tags allow, for example, differentiating between facts that represent passwords and user names.

We assign unique tags to messages with the function $Tag\colon \mathcal{T}_\Sigma(X) \to \mathcal{T}_{\mathcal{C}_{pub}\cup\{\mathrm{pair}\}}(X)$, which is a homomorphism between $\mathcal{T}_\Sigma(X)$ and $\mathcal{T}_{\mathcal{C}_{pub}\cup\{\mathrm{pair}\}}(X)$ considered as semigroups with respect to the (associative) pairing function. An explicit construction of the tag function is given in Appendix C. We add tags to all agent rules that contain a send or receive event to or from a human role. The tag assignment is based on the representation of a message in the human's role script. In the following, we write *'varname'* to denote $Tag(varname)$. To illustrate the tags, imagine that we modify the protocol *SimpleProtocol*, Figure 2, such that S communicates with a human H instead of R. For example, for Rule (S3) in Figure 4, we introduce the tag *'m_2'* as follows:

$$[\mathsf{AgSt}(S, 2, \langle H, m_1\rangle), \mathsf{In}_{sec}(\langle H, S, \langle \text{`}m_2\text{'}, m_2\rangle\rangle)] \xrightarrow{\texttt{Receive}(S,sec,H,\langle \text{`}m_2\text{'},m_2\rangle)} [\mathsf{AgSt}(S, 3, \langle H, m_1, m_2\rangle)]$$

The untrained-human rules formalize arbitrary behavior of fallible humans. They define that in every system state the human's current knowledge can be sent to any agent, including the adversary, over any available channel. Similarly, a term can be received from anyone and stored in the human's knowledge.

To keep track of the human's knowledge, we introduce persistent facts for every message in the human's knowledge. $!\mathsf{HK}(H, \langle t, x\rangle)$ denotes that the human H knows the message x, tagged with t. Figure 5 depicts three untrained-human rules. At any time, a human can produce a new fresh value x and store it in his knowledge together with a tag t. This is modeled by Rule (HR1). The premise $\mathsf{Fr}(x)$ of the rule can be produced in any system state with the fresh rule (F1). A human can send any message from his knowledge to the network as well as receive any new message from the network and store it in his knowledge. Rules (HR2) and (HR3) respectively model the sending and receiving of a message over an insecure channel. The model contains analogous rules for the sending and updating of the human's knowledge over authentic, confidential, and secure channels (see Appendix B). To initialize an untrained human H, we provide a setup rule that produces for every

$$[\mathsf{Fr}(x)] \xrightarrow{\texttt{Fresh}(H,\langle t,x\rangle)} [!\mathsf{HK}(H, \langle t, x\rangle)] \quad \text{(HR1)}$$

$$[!\mathsf{HK}(H, \langle t, x\rangle)] \xrightarrow{\texttt{Send}(H,ins,P,\langle t,x\rangle)} [\mathsf{Out}_{ins}(\langle H, P, \langle t, x\rangle\rangle)] \quad \text{(HR2)}$$

$$[\mathsf{In}_{ins}(\langle P, H, \langle t, x\rangle\rangle)] \xrightarrow{\texttt{Receive}(H,ins,P,\langle t,x\rangle)} [!\mathsf{HK}(H, \langle t, x\rangle)] \quad \text{(HR3)}$$

Fig. 5: Untrained-human rules for producing fresh messages, sending messages to and receiving messages from an insecure channel.

message x in H's initial knowledge a fact $!\mathsf{HK}(H, \langle t, x\rangle)$ and a corresponding action $\texttt{InitK}(H, \langle t, x\rangle)$.

Similarly to the agent rules, every untrained-human rule is labeled with an action $\texttt{Start}(H, \langle t, x\rangle)$, $\texttt{Fresh}(H, \langle t, x\rangle)$, $\texttt{Send}(H, l, P, \langle t, x\rangle)$, or $\texttt{Receive}(H, l, P, \langle t, x\rangle)$. H denotes the human agent performing the action, l the channel type, and P the apparent communication partner. In contrast to actions not concerning human roles, the last argument not only denotes the message x, but also its tag t.

B. Infallible and Untrained Humans

Given the agent rules and the untrained-human rules, we next present how we realize infallible and untrained humans. The infallible human follows the role specification and is therefore modeled by the agent rules. That is, except for the fact that all messages are tagged, he is modeled in the same manner as all non-human agents. This (unrealistically) strong model is how human agents are generally modeled.

The untrained human has no knowledge about the protocol and can blindly follow all instructions given to him. He can thereby deviate arbitrarily from the protocol specification. The untrained human agent is therefore modeled by the untrained-human rules without further restrictions. As a result, no protocol can ensure the secrecy of messages that the untrained human knows because the adversary can always query the human for these messages, e.g. using social engineering.

C. Rule-Based Humans

A rule-based human agent is defined by the guidelines he follows. An example of such a guideline is "private keys must be kept secret." Guidelines are simple statements that are expected to be remembered and followed by humans in practice. For example, it is unrealistic for a bank to assume that all customers know every step of an e-banking procedure. However, many service providers require in their terms of service that their customers do not reveal their passwords to other parties.

Guidelines state what the human must or must not do and therefore restrict how the human can deviate from the protocol specification. A rule-based human is formally modeled by the untrained-human rules together with predicates that exclude all traces from consideration that do not satisfy the predicates.

$$NoTell(H, tag) := \forall \texttt{Send}(H, l, P, \langle t, m\rangle) \in tr, t', m' : \langle t, m\rangle \vdash_H \langle t', m'\rangle \Rightarrow t' \neq tag$$

$$NoTellExcept(H, tag, rtag) := \forall \texttt{Send}(H, l, P, \langle t, m\rangle) \in tr, m', R : \texttt{InitK}(H, \langle rtag, R\rangle) \in tr \wedge \langle t, m\rangle \vdash_H \langle tag, m'\rangle \Rightarrow P = R \wedge (l = \mathit{sec} \vee l = \mathit{conf})$$

$$NoGet(H, tag) := \forall \texttt{Receive}(H, l, P, \langle t, m\rangle) \in tr, t', m' : \langle t, m\rangle \vdash_H \langle t', m'\rangle \Rightarrow t' \neq tag$$

$$ICompare(H, tag) := \forall \texttt{Receive}(H, l, P, \langle t, m\rangle) \in tr, m' : \langle t, m\rangle \vdash_H \langle tag, m'\rangle \Rightarrow \texttt{InitK}(H, \langle tag, m'\rangle) \in tr$$

Fig. 6: Predicates modeling guidelines of a rule-based human.

Such predicates are expressed as axioms in the Tamarin tool. While one can envision all kinds of guidelines and formalize the corresponding predicates, in this paper we consider only four exemplary predicates.

The four predicates we define are relevant for several reasons. First, they concern three capabilities that are assumed by standard agents in communication protocols: sending, receiving, and comparing messages. Second, the predicates express conditions that must be satisfied in every protocol step rather than a particular protocol step. We prefer such predicates because they correspond to simpler guidelines.

In the following, we write $\langle t, m\rangle \vdash_H \langle t', m'\rangle$ if a human agent can select the term $\langle t', m'\rangle$ in $\langle t, m\rangle$. Formally, $\langle t, m\rangle \vdash_H \langle t', m'\rangle \Leftrightarrow \exists i, k : 1 \leq i \leq k \wedge t = \langle t_1, \ldots, t_k\rangle \wedge m = \langle m_1, \ldots, m_k\rangle \wedge t' = t_i \wedge m' = m_i$.

Our first predicate, $NoTell(H, tag)$, specifies traces in which the human H does not send information of type tag to anyone. This is relevant for achieving message secrecy. For example, the guideline that private keys must be kept secret is expressed as $NoTell(H, \textit{'private key'})$. The formal definition of the predicate $NoTell(H, tag)$ is shown in Figure 6. It states that the human H does not send a message $\langle t, m\rangle$ that contains a subterm m' with the tag tag. Note that all the predicates in Figure 6 have at least two arguments: the human agent H that adheres to the corresponding guideline and the tag tag referring to the type of message that must or must not be used in the specified manner.

Passwords are another type of information that should not be publicized. However, prohibiting their communication with the $NoTell$ predicate is inappropriate for authentication protocols that require the human to input his password for authentication. Humans that adhere to the corresponding guideline cannot successfully complete such protocols. We thus refine the guideline to require that the human can only send his password to a designated agent over a confidential or secure channel, but not to anyone else. The corresponding predicate, $NoTellExcept(H, tag, rtag)$, states that if the human H associates an agent R with tag $rtag$ in his initial knowledge, he never sends a message containing the tag tag to anyone except to R. Additionally, H can only send this message on a confidential or secure channel. In the predicate's formal definition, shown in Figure 6, we make use of the action fact `InitK` that denotes a term in the human's initial knowledge.

The predicate $NoGet(H, tag)$, shown in Figure 6, states that the human H rejects any message that contains the tag tag. In particular, if a human H rejects a message, this message cannot update the human's knowledge with a fact that originates from another agent. The corresponding guideline helps protect the integrity of some types of information. A common attack, for instance, is to disguise malicious software as a security update of a popular software package. A safe update method is to use the software's built-in update mechanism. Humans should therefore only use the built-in mechanism and not follow a suggested link to an updater. This can be expressed with the predicate $NoGet(H, \textit{'update link'})$.

Humans are known to skip verification steps. A human asked to confirm a transaction is likely to proceed without paying close attention to details. Stating a guideline that requires the human to pay attention is not a satisfactory solution. A simple technique to make a human pay more attention is to ask him for input and afterward provide him with information that must be verified along with a random code used for confirmation. Some humans may of course still ignore the comparison and simply enter the code, but others will not. Our final predicate distinguishes between these two types of humans. The $ICompare(H, tag)$ predicate states that whenever the human H receives a message with tag tag, he compares it to the message associated with the same tag in his initial knowledge. The effect of this predicate, in combination with the untrained human rules, is that the human agent either ignores the entire message or verifies the tagged subterm.

Summarizing, these four predicates allow us to model human agents that never send or receive a message, or always perform certain comparisons. Furthermore, we have illustrated on the example of $NoTell$, that the corresponding guidelines can be made more specific when necessary. The $NoTellExcept(H, tag, rtag)$ predicate describes the same property as $NoTell(H, tag)$ but allows us to make an exception for a trusted agent associated with the tag $rtag$. In a similar manner, the other guidelines could be made more specific. However, as we will show in the next sections, humans who follow the rules introduced here are sufficiently strong to avoid many attacks.

D. Errors of other Fallible Humans

We have introduced the means to formalize all roles appearing in the protocol specification. In addition, we allow for an arbitrary number of untrained human agents. We thereby single out a distinguished human agent for which we examine the security properties. This distinguished human can be an

0. H : $\text{knows}(D, P, S, pw, idH, idS)$
0. D : $\text{knows}(H, idH, \text{pk}(skS))$
0. S : $\text{knows}(skS, H, idS, pw, idH)$
1. H $\bullet\!\!\rightarrow\!\!\bullet$ P : S
2. P $\circ\!\!\rightarrow\!\!\bullet$ S : *'start'*
3. S $\bullet\!\!\rightarrow\!\!\bullet$ P : $\text{fresh}(rS).idS, rS$
4. P $\bullet\!\!\rightarrow\!\!\bullet$ D : idS, rS
5. D $\bullet\!\!\rightarrow\!\!\bullet$ H : idS
6. H $\bullet\!\!\rightarrow\!\!\bullet$ D : pw, idH
7. D $\bullet\!\!\rightarrow\!\!\bullet$ P : $\text{fresh}(rD).\text{aenc}(rD, \text{pk}(skS)),$ $\text{senc}(\langle f(rS), idH, pw\rangle, f(rS, rD))$
8. P $\bullet\!\!\rightarrow\!\!\bullet$ S : $\text{aenc}(rD, \text{pk}(skS)),$ $\text{senc}(\langle f(rS), idH, pw\rangle, f(rS, rD))$
9. S $\bullet\!\!\rightarrow\!\!\bullet$ P : $\text{senc}(f(rD), f(rS, rD))$
10. P $\bullet\!\!\rightarrow\!\!\bullet$ D : $\text{senc}(f(rD), f(rS, rD))$
11. D $\bullet\!\!\rightarrow\!\!\bullet$ H : *'success'*

Fig. 7: *MP-Auth* protocol

instantiation of any human definition. The additional human agents that we allow for are always untrained. This models that an adversary can trick as many humans as he wants into blindly following his instructions when this helps him to attack the distinguished human. Our model is thus a Dolev-Yao model with untrained humans and a distinguished test human that may be infallible, untrained, or guided by one or more of the rules that we introduced.

V. Case Study in Protocol Analysis

To validate the utility of our approach, we illustrate how it can be used to automatically find attacks that arise from human errors. We first present the authentication protocol *MP-Auth* [18] and then analyze whether it provides entity and message authentication from a human to a remote server. In the next section, we compare it to five other protocols that have the same aim, but are based on different mechanisms.

MP-Auth authenticates a human H to a server S using the human's platform P and his personal device D, to which the human has exclusive access. A simplified version of this protocol, obtained by merging H and D into a single role, was shown to provide mutual authentication between D and S and to establish a secret symmetric session key between these two parties [18].

The main idea of the protocol, shown in Figure 7, is that a human never needs to enter his password on the platform that may be controlled by an adversary. The device D has the public key of the server S preinstalled. We first explain the protocol in detail and then analyze it with respect to entity and message authentication. Afterward, we present a version of the protocol that incorporates a method to harden protocols against human errors and we analyze its effectiveness.

The protocol (Figure 7) proceeds as follows: in Step 1, the human enters the name of the server he wants to communicate with on the platform P. P then initiates communication with the server S (Step 2). Next, S produces a fresh value rS and sends this, together with its identity information idS via the platform P to device D (Steps 3 and 4). We assume that the short-range network connection between D and P is secure. In Step 5, D displays idS to the human, who checks if this corresponds to the server he wants to communicate with. If it does, he enters his password pw and identity idH on the device. Upon receiving this message, the device produces a fresh nonce rD and encrypts it with the public key of S. D applies the hash function f to rS and concatenates it with idH and pw. Then, D encrypts the concatenated message with $f(rS, rD)$ and the two encrypted messages are sent together to the platform P, which relays them to the server S (Steps 7 and 8). The last three steps authenticate the server S to the human H. They are not relevant for entity authentication from H to S but will be relevant for message authentication in the next section. In Steps 9 and 10, S applies f to the received rD and sends it, encrypted with $f(rS, rD)$, over P back to D. D checks if rD corresponds to the nonce that it has previously sent and indicates *'success'* to the human if it does.

We model that D is preconfigured with the public key of S by including this key in D's initial knowledge. Furthermore, we assume the existence of secure channels between H and P as well as between H and D. The first connection from P to S is confidential, while the following communication is secure. This models a TLS connection between the two parties.

In the following analysis, we make the realistic assumption that the platform P is under the adversary's control, i.e., corrupt. We assume that the human only launches applications on a malware-free device D and therefore do not consider D to be corrupt. We model P by omitting its agent rules from the specification. Instead, we model every message that is either sent or received by P as being received from or sent to the insecure network that is controlled by the adversary.

A. Analysis

We analyze the *MP-Auth* protocol with the Tamarin tool with respect to both an infallible and an untrained human. The infallible human serves as a best case scenario: If a protocol does not satisfy a property with an infallible human, it will also not satisfy the property with an untrained human. Whenever a property holds for an infallible human, but not for an untrained one, we also examine what rules must be known by a rule-based human for the property to be satisfied. Whenever a property is proven by Tamarin, we only state this and refer to the Tamarin files [6] for the detailed specifications, assumptions, and proofs.

We first analyze if entity, device, and message authentication hold in the *MP-Auth* protocol. Afterward, we suggest improvements to the protocol and analyze their effectiveness.

1) Entity and Device Authentication: First, we examine if entity authentication and device authentication hold for the different kinds of human agents. The assumption for device authentication is that the human always carries the device D on him and has exclusive access to it.

Claim 1. MP-Auth *provides entity authentication and device authentication from the human H to the server S for an infallible human but neither property for an untrained human.*

Proof. For an infallible human, the two properties are proven by Tamarin. In contrast, for an untrained human Tamarin finds attacks for both properties, which can be interpreted as follows: the untrained human can enter his password on the corrupted platform before S is active, even if this is not foreseen in the specification. An adversary can thus learn the password and therefore generate all messages that S expects (Steps 2 and 8 in Figure 7). This attack is possible in a trace where the human is only active before the first action of S and the human's device is not active at all. Consequently, both entity and device authentication fail. □

We therefore examine a rule-based human, who knows that he must send passwords, i.e., messages tagged with 'pw', only to the device 'D', to which he has a secure channel, and not to anyone else. We use Tamarin to establish the following:

Claim 2. MP-Auth *provides entity authentication, as well as device authentication, from the human H to the server S for a rule-based human who knows he can only enter the password on the device to which he has a secure channel, expressed by $NoTellExcept(H, \text{'pw'}, \text{'D'})$.*

2) Message Authentication: [18] describes additional steps to authenticate transactions, which are depicted as Steps 12–19 in Figure 8. After the login phase, a human H can enter his message m on the platform P, which relays it to the server S. S produces a fresh nonce rS_2 and encrypts it and the message m with the key $f(rS, rD)$. Then S sends the result over P to the device D which can decrypt it. D displays m to H and waits for his confirmation, OK, that this is the message that H previously sent. Then D computes the function f over rS_2 and m, encrypts it, and sends it over P back to S.

The final protocol for sending a message m authentically from H to S consists of the Steps 1–11 in Figure 7 followed by the Steps 12–19 in Figure 8. We examine whether the second part of the protocol, which we call *MP-Auth_MA*, satisfies the message authentication property. However, we take into account that at the time when m is sent there has already been a login protocol. Therefore, the communicating agents already share some knowledge at the beginning of *MP-Auth_MA*. Next, we justify why we can assume this shared initial knowledge.

Analysis with Tamarin shows that with an infallible human, after the first part, D and S agree on the value of H and on the value of their shared key $f(rS, rD)$. More specifically, S can be sure to share these messages with another agent, but does not necessarily know with which other agent. This is because S never learns what device D is participating in the same protocol. However, D can be sure with which server S it has an agreement. We call this security property a *weak data agreement* between D and S and refer to Definitions 5 and 6 of Appendix A for its formal definition. Further examination shows that the same property also holds for a rule-based human who knows that he can only send his password to the device. We thus assume that the human was at least behaving according to this rule in the first part and analyze

0.		H :	$\text{knows}(D, P, S, m, idH, idS, OK)$
0.		D :	$\text{knows}(H, idH, \text{pk}(skS), idS, S, rS, rD)$
0.		S :	$\text{knows}(skS, H, idS, idH, rS, rD)$
12.	H $\bullet\!\!\rightarrow\!\!\bullet$	P :	m
13.	P $\bullet\!\!\rightarrow\!\!\bullet$	S :	m
14.	S $\bullet\!\!\rightarrow\!\!\bullet$	P :	$\text{fresh}(rS_2).\text{senc}(\langle m, rS_2\rangle, f(rS, rD))$
15.	P $\bullet\!\!\rightarrow\!\!\bullet$	D :	$\text{senc}(\langle m, rS_2\rangle, f(rS, rD))$
16.	D $\bullet\!\!\rightarrow\!\!\bullet$	H :	m
17.	H $\bullet\!\!\rightarrow\!\!\bullet$	D :	OK
18.	D $\bullet\!\!\rightarrow\!\!\bullet$	P :	$\text{senc}(f(m, rS_2), f(rS, rD))$
19.	P $\bullet\!\!\rightarrow\!\!\bullet$	S :	$\text{senc}(f(m, rS_2), f(rS, rD))$

Fig. 8: *MP-Auth_MA* protocol

MP-Auth_MA with the shared initial knowledge of D and S as depicted in Figure 8. Consequently, we use the term "untrained-human" here to denote a human who behaves arbitrarily in the new part of the protocol only.

Claim 3. MP-Auth_MA *provides message authentication of m from the human H to the server S for an infallible human, but not for an untrained one.*

Proof. Message authentication for an infallible human is proven by Tamarin. In contrast, Tamarin finds an attack with an untrained human that can be interpreted as follows: An adversary can send his own m' to the server S, which will therefore be displayed by D. However, an untrained human presses OK without reading the display, and thereby confirms the message m' from the adversary. □

To avoid this attack, a human must not press OK unless he has received his message m from the device in the previous step. However, this cannot be expressed with the set of guidelines defined in Section IV-C and we show in Appendix D that the *MP-Auth_MA* protocol does not provide message authentication for any combination of these guidelines.

The above attack is possible because the human does not read his device's display. If he is forced to read the display to proceed in the protocol, then he can no longer ignore it. Next, we analyze an improved version of the protocol that incorporates this idea.

B. MP-Auth_VC

We call the improved protocol *MP-Auth_VC*, which incorporates a verification code, and we explain it on an example. Imagine that the human H wants his bank server S to carry out a transaction m. We model m to be in H's initial knowledge. In practice, H could remember the transaction information by writing the details down on paper.

Remark. By modeling the message m in the human's knowledge, we assume that the information noted on paper reflects the human's intention and that he will not misinterpret it. △

In the original protocol in Figure 8, the device displays in Step 16 all transaction details m, and the human confirms them by pressing an OK button in Step 17. We suggest that Steps 16 and 17 are replaced by the following two steps.

16. D $\bullet\!\!\rightarrow\!\!\bullet$ H : $\mathrm{fresh}(vc).vc, m$
17. H $\bullet\!\!\rightarrow\!\!\bullet$ D : vc

Instead of displaying just the transaction details, the device additionally displays "if this is your message, please confirm by entering the following verification code." The code should be fresh and unpredictable in every protocol run. We model this by Step 16 above, where D sends m together with a fresh verification code vc to the human. The protocol only proceeds if the human enters vc on the device, as in Step 17. A human in this protocol cannot ignore what is displayed on the device and just press OK. Instead, he is forced to read the display to learn vc. Nevertheless, there is an attack possible with an untrained human.

Claim 4. MP-Auth_VC *provides message authentication of m from a human H to a server S for an infallible human, but not for an untrained one.*

Proof. Tamarin proves message authentication for an infallible human. Tamarin finds an attack for an untrained human, which can be interpreted in the context of our transaction example as follows: An adversary can trigger his own transaction m'. Consequently, the corresponding transaction details are displayed by the device D. However, an untrained human ignores that m' should represent his transaction details. Therefore, he wrongly enters vc on the device and confirms the adversary's transaction. □

To avoid this attack, we consider a rule-based human who checks at each step where he receives a message m with the tag *'m'*, whether this corresponds to the message associated with *'m'* in his initial knowledge. This is expressed by the guideline $ICompare(H, \text{'m'})$. In our example, this means that whenever the human reads on the display a transaction, he checks that the corresponding information is the transaction information he previously noted on paper. Tamarin establishes the following claim.

Claim 5. MP-Auth_VC *provides message authentication of m from a human H to a server S for a rule-based human who knows that whenever he receives a message containing x with tag 'm', he must check if x corresponds to the message associated with 'm' in his initial knowledge, expressed by $ICompare(H, \text{'m'})$.*

The improved protocol is thus secure with respect to a rule-based human who performs certain checks. This is because, unlike the original version, it is not possible for the human to ignore the displayed information altogether.

C. Conclusion from Case Study

We have demonstrated that we can use our model to automatically find realistic attacks that are caused by human errors. We considered untrained, rule-based, and infallible humans and showed that entity and device authentication do not hold with untrained humans but hold with rule-based ones.

When considering message authentication, we have seen how attacks arise when humans do not perform certain checks. We have therefore suggested an improved version of the protocol that enforces such a check. Our analysis shows that this improvement is only effective with a rule-based human. Our model can thus help to find ways to avoid human errors by improving the protocol specifications and the guidelines that humans must follow.

The argument that one protocol improves another is dependent on the set of guidelines we chose in our model. We posit that our four rules provide a good basis for analyzing protocols. They model general guidelines that express what a human must or must not do in every moment. In reality it is expected that humans behave according to such guidelines, for example that they perform basic checks.

VI. Comparing Authentication Protocols

In this section, we use our model to make fine-grained distinctions between protocols that appear equally secure in the absence of human errors. We compare six mobile phone-based authentication protocols that all aim to authenticate a human to a server but use different mechanisms to achieve this goal.

In addition to *MP-Auth*, we examine the protocols *Cronto* [1], *Google 2-Step* [2], *OTP over SMS*, and *Phoolproof* [22]. These protocols were compared by *Bonneau et al.* [8] using a framework to analyze protocols designed to replace password-based authentication. The framework compares how usable, deployable, and secure the protocols are. We also consider the recently proposed protocol *Sound-Proof* [16] that is also phone-based. We analyze whether each of the protocols provides entity, device, and message authentication from the human H to the server S.

In each protocol, a human H wants to authenticate himself to a server S. For this purpose, he has access to a platform P, through which he communicates with the server, and he also has exclusive access to a personal device D, like a cell phone. As in the previous section, where not stated otherwise, we assume that the channels between H and P as well as between H and D are secure, that the first connection from P to S is confidential, and that the subsequent communication between P and S is secure (modeling TLS). Also like in the previous section, we assume that the platform is under the adversary's control. Again, we analyze the protocols using Tamarin and consider an infallible and an untrained human. A rule-based human is considered when the results for the infallible and untrained human differ.

First, we explain why we examine both entity and message authentication in all protocols. Afterward, we present the protocols and our results.

A. Entity and Message Authentication

Normally it is a small step to achieve message authentication when entity authentication holds. However, when we take human errors into consideration, this step is no longer straightforward. For this reason, it is necessary to also examine message authentication.

Message authentication is often easily achieved when an entity has already been authenticated that is associated with a key. In a setting with two non-human entities A and B, A can encrypt every message with the previously authenticated key that B knows belongs to A. However, this no longer works when a human agent H is authenticated because end-to-end encryption is not possible. Even if a server S has correctly authenticated H, H can leave his platform and another human H' can take over and send commands to S. As the platform P is an intermediate party between H and S, S has no means of detecting this if additional authentication is not done. This problem has already been considered in the context of many password change mechanisms: When a user is already logged in, he must reenter his password prior to setting a new one. This prevents someone else from changing the password in his name, for example during his absence from the platform.

Remark. The platform P could continually check if the human is still present, for example using a camera or similar techniques. However, with this approach, the hardware making this check must be trusted and function error-free. In particular, we consider a corrupted platform. In this case, letting the platform check whether the user has changed does not help the server. $\triangle$

From the discussion above we see that even after a successful login, the server must check if message authentication is given after every transaction, or after a batch of transactions.

B. Protocols and Results

We start by briefly introducing the protocols. We refer to Section V for the description of *MP-Auth* and to Appendix E for more details and the specifications of the other protocols.

Cronto is a challenge-response protocol, where the server encrypts a fresh nonce in the form of a cryptogram, a graphical representation of a cipher text, that is displayed on the human's platform. The human uses his auxiliary device to scan the cryptogram. The device then decrypts it and computes a one-time password from the extracted information that is displayed to the human. The human must enter the one-time password, together with his normal password, on the platform.

The *Google 2-Step* protocol models Google's two-factor authentication. After the human enters his identity and password, he receives from the server a fresh code on a second channel. For example, he receives an SMS, i.e., a text message, that is sent to his mobile phone. The human must then enter this code on the platform.

Similarly, *OTP over SMS* is a challenge-response protocol where the server sends a one-time password (OTP) to the human over his device. The one-time password that the human then enters on the platform is a single factor for authentication.

Phoolproof is the only protocol we examine that does not start with the step where the human tells his platform to which server he wants to connect. This avoids the human directing his platform to connect to a corrupt server. Instead, the human selects on his device the server from a list of registered servers and the device communicates this choice to the platform. The device and the server then authenticate each other by a signed challenge-response mechanism. Only after this does the human enter his password on the platform to log in.

Sound-Proof is based on the idea that something that identifiably belongs to the human, namely his device, must be in close proximity to the platform used by the human. Whether the device and platform are actually near each other is decided by measuring ambient noise. If the noise measured by both the platform and the device are roughly the same, then they are deemed to be at the same location. The comparison of the two recordings is performed by the device.

	Entity Auth.			**Device Auth.**			**Message Auth.**		
	I	*U*	*R-B*	*I*	*U*	*R-B*	*I*	*U*	*R-B*
Cronto	✓	✓		✓	✓				
Cronto_MA							✓	×	✓(2)
Google 2-Step	✓	✓		✓	✓		✓	×	✓(2)
MP-Auth	✓	×	✓(1)	✓	×	✓(1)			
MP-Auth_MA							✓	×	×
MP-Auth_VC							✓	×	✓(2)
OTP over SMS	✓	✓		✓	✓		✓	×	✓(2)
Phoolproof	×	×		✓	✓		✓	✓	
Sound-Proof	×	×		✓	✓		×		

(1) with rule $NoTellExcept(H, \text{'pw'}, \text{'D'})$
(2) with rule $ICompare(H, \text{'m'})$

TABLE I: Entity authentication (Entity Auth.), device authentication (Device Auth.), and message authentication (Message Auth.) of human H to server S, with an infallible (*I*), untrained (*U*), and rule-based (*R-B*) human.

Table I shows the results established with Tamarin. The three parts indicate whether the protocols satisfy the properties entity authentication, device authentication, and message authentication from the human to the server, for an infallible human (*I*) and for an untrained one (*U*). Whenever the property holds for an infallible but not for an untrained human, we examine if it holds for a rule-based human (*R-B*). A tick ✓ in the table means that the respective authentication property from H to S holds, and × indicates that it fails. The numbers following the ticks reference what rule must be known by the rule-based human, listed at the table's bottom. Next, we present the most interesting results.

1) Entity and Device Authentication: We first discuss which protocols provide entity authentication. Tamarin establishes:

Claim 6. Cronto, Google 2-Step, *and* OTP over SMS *provide entity authentication from a human H to a server S for an infallible human and for an untrained human.*

Entity authentication holds in these protocols because the human must read from his device the one-time password or fresh code from the server and enter it on the platform.

Claim 7. Sound-Proof *and* Phoolproof *neither provide entity authentication from a human H to a server S for an infallible human nor for an untrained human.*

Proof. Tamarin finds attacks for both protocols with an infal-

lible as well as with an untrained human. □

We present an interpretation of the attacks with infallible humans. This attack also works, of course, for untrained humans. In *Sound-Proof*, even if the protocol proceeds as intended, the human H is only active before the first event of the server S. This participation need not, however, be recent (as defined in Section III-D). In *Phoolproof*, entity authentication fails because the adversary can learn the human's password when he enters it on a corrupted platform in a protocol run prior to the moment when S starts. The password can then be replayed by the adversary at the point of the protocol where S expects to receive it, and thus H need not be active between the start and end of S.

Since entity authentication fails, we also examine device authentication (see Section III-D) for these two protocols. For device authentication, we require that a human H has exclusive access to his device D, which he always carries on him. We prove with Tamarin:

Claim 8. Sound-Proof *and* Phoolproof *provide device authentication of H to S for an infallible and an untrained human.*

Intuitively, *Phoolproof* satisfies this property because H's device D must participate in the protocol to send the signed challenge back to S. Similarly, it holds in *Sound-Proof* because the device D communicates over a secure channel to S that the two recordings are equal.

2) Message Authentication: A variation of the *Cronto* protocol can be used for transaction authentication. As it is not clear from the protocol's description whether this can only be done after a login, we examine it as a separate protocol without login. We name this variation of the protocol *Cronto_MA* and examine whether it provides message authentication. In this version of the protocol, the one-time password is based on both a fresh nonce and the message sent by the human. Also, no password, in addition to the one-time password, must be entered by the human. All remaining protocols do not specify how to authenticate a message m. Nevertheless, as message authentication is important, we examine if the protocol steps can be enhanced to provide this property.

For this purpose, we introduce a fresh message m that is only known by the human H at the protocols' start and extend the protocols with m using the results of *Basin et al.* [5] on human-interaction security protocol (HISP) topologies. HISP topologies consist of four parties: an infallible human, his personal device, the human's platform, and a server. The human can access his device and his platform, through which he can communicate with the server. The assumption is that the platform is always corrupt. This setup matches with all our protocols when we consider infallible humans.

We use the results of [5] as follows. Minimal HISP topologies state necessary conditions for message authentication. Thus, for each of the protocols, we must add the message m to a set of communication channels that covers one of the minimal topologies. Otherwise, we know that it is impossible to achieve message authentication even with an infallible human. We refer to Appendix E for the resulting protocols in Alice&Bob specification.

Claim 9. *It is impossible to use* Sound-Proof *for message authentication with an infallible human and a corrupted platform.*

Proof. It is not possible to cover a minimal HISP topology in this protocol, even if m is added to every message. By Theorem 2 of [5], it follows that with this protocol it is impossible to achieve message authentication with a corrupted platform. □

Claim 10. Cronto_MA*,* Google 2-Step*, and* OTP over SMS *provide message authentication of m from a human H to a server S for an infallible human, but not for an untrained human.*

Proof. Tamarin proves message authentication for an infallible human. Furthermore, in all three protocols, Tamarin finds attacks for an untrained human, which can be interpreted as follows: An adversary can send his own message m' to the server. Consequently, m' will be displayed on the device together with the one-time password in *Cronto_MA* and *OTP over SMS*, and together with the fresh code in *Google 2-Step*. An untrained human reads this challenge from his device and enters it on the platform without checking the corresponding message. With this, he confirms any message m' of the adversary. In *Google 2-Step*, the adversary additionally must know the password, which he learns when the human enters it on the corrupt platform. □

We next consider these protocols with a rule-based human who knows that every time he reads information about a message with tag *'m'*, he must check if this message is the one associated with *'m'* in his initial knowledge. This means that a human will only copy the server's challenge from the device to the platform if it is sent together with the transaction that he intended to make. We establish with Tamarin:

Claim 11. Cronto_MA*,* Google 2-Step*, and* OTP over SMS *provide message authentication of m from a human H to a server S for a rule-based human who knows that whenever he receives a message containing x with tag 'm', he must check if x corresponds to the message associated with 'm' in his initial knowledge, expressed by $ICompare(H, \text{'m'})$.*

We next show with Tamarin that the protocol *Phoolproof* is the only protocol that achieves message authentication even with an untrained human.

Claim 12. *The* Phoolproof *protocol provides message authentication of m from a human H to a server S for an infallible as well as for an untrained human.*

The human not only enters the name of the server S but also the message m on the device D. D then signs S and m before sending them over the corrupt platform P to S. The adversary cannot forge D's signature and therefore cannot alter the message m that is sent by the human.

C. Discussion

Our case studies show that we can use our model and automated reasoning support to find realistic attacks arising from human errors. We can also use them to compare the resilience of protocols to such attacks, where the protocols exploit different defense mechanisms. Table I shows that three of the six protocols analyzed (*Cronto*, *Google 2-Step*, and *OTP over SMS*) provide entity authentication even with untrained humans. The protocols *Phoolproof* and *Sound-Proof* provide device authentication with untrained humans. *MP-Auth* provides entity and device authentication only with rule-based humans.

Phoolproof is the only protocol that provides message authentication with untrained humans. At the other extreme, *Sound-Proof* cannot be used for message authentication at all. All other protocols, except *MP-Auth_MA*, provide message authentication for rule-based humans, but not for untrained ones. In all of these protocols, the human must check whether a displayed message is one that he has previously sent. We have proposed an improved version of *MP-Auth_MA* that forces a human to read the display. However, as we have observed, untrained humans can still fail to compare the relevant parts.

The *Cronto* and *Phoolproof* protocols have been rated equal in all security categories by *Bonneau et al.* [8]. By taking human errors into account, we have further differentiated them: While *Cronto* provides entity authentication, *Phoolproof* only provides device authentication. However, if *Phoolproof* was used for message authentication, it would be secure even with untrained humans. In contrast, the variation *Cronto_MA* only satisfies message authentication with rule-based humans.

VII. Related Work

Smith [26] observed over a decade ago that although the opinion is widely held that many security problems are caused by the interaction of humans with computers, the analysis and design of security protocols usually focus on just the computer parts. Today, human-computer interaction is an active research area, also in the domain of Information Security. We focus here mainly on related work concerned with formal models for Information Security that include fallible humans.

Carlos et al. [9] model the human formally as part of a security ceremony [14] in which a protocol is analyzed. They introduce a human agent that can store knowledge and send and receive messages over different media corresponding to either human-to-human or human-to-machine communication. This is similar to the untrained human that we consider, in that we also model the human's knowledge and rules used to send and receive new knowledge facts. While we use the same channels for all communication, [9] distinguishes the events communicated over different media. It is unclear how their framework can be instantiated with an actual security protocol. Moreover, although they provide a formal human model, human errors are not explicitly covered.

Rukšėnas et al. [24] propose a formal human model that allows for human errors. Their focus is on finding errors that stem from the design of the interface between a standard human user and a system. They do not consider communication protocols. Rather, they consider scenarios where the interaction between one system, one human (and his interpretation of the system), and one specific adversary is analyzed.

Beckert and Beuster [7] share a similar viewpoint to [24] in that they also examine scenarios with human-machine interfaces. They give a formal semantics to a known psychological human model, a variation of the GOMS model [15], and also model the user's assumptions about the application. The resulting model of human-computer interaction consists of the human model, the user's assumptions, and a formal application model. When reasoning about human errors, they use what we called the skilled human model in Section II. That is, they explicitly model that a user can behave incorrectly or that he can misinterpret the system's state.

Both *Rukšėnas et al.* [24] and *Beckert and Beuster* [7] only consider scenarios with a fixed number of agents. In contrast, our model allows arbitrary many instantiations of the roles and of untrained humans and supports unbounded verification.

VIII. Conclusion

We have formalized security protocols with fallible humans, mechanized their analysis within the Tamarin tool, and conducted case studies on various authentication protocols. We demonstrated that our model can be used to analyze what guidelines must be given for humans to securely execute a protocol. Such insights can serve as a starting point to prioritize the security information that must be given to non-expert users in practice. We also showed that our model allows us to make fine distinctions between protocols that were previously considered equally secure. The consideration of human errors therefore adds a new dimension to the classification of security protocols.

There are several directions for future work and we highlight two of them. First we have formalized guidelines for rule-based humans that have proven useful for our case studies. Nevertheless, other guidelines are conceivable. However, to develop and compare guidelines, we must first find useful evaluation criteria.

Second we have presented two approaches for human error analysis based on skilled and rule-based humans. However, most humans do not neatly fit into one of these types. A more realistic human error model should consider the combination of the skilled and rule-based approaches. However, such a model dramatically increases the number of possible system behaviors and we must therefore find strategies to cope with the resulting verification complexity.

References

[1] Cronto. http://www.cronto.com/.

[2] Google 2-step. https://www.google.com/landing/2step/.

[3] Martín Abadi and Cédric Fournet. Mobile Values, New Names, and Secure Communication. *SIGPLAN Not.*, 36(3):104–115, January 2001.

[4] Mohamed Alsharnouby, Furkan Alaca, and Sonia Chiasson. Why phishing still works: User strategies for combating phishing attacks. *Int. J. Hum.-Comput. Stud.*, 82:69–82, 2015.

[5] David Basin, Saša Radomirović, and Michael Schläpfer. A Complete Characterization of Secure Human-Server Communication. In *28th IEEE Computer Security Foundations Symposium (CSF 2015)*. IEEE Computer Society, 2015.

[6] David Basin, Saša Radomirović, and Lara Schmid. Modeling Human Errors in Security Protocols (Full Version). Available with Tamarin specification files at http://www.infsec.ethz.ch/research/projects/hisp.html.

[7] Bernhard Beckert and Gerd Beuster. A method for formalizing, analyzing, and verifying secure user interfaces. In *Formal methods and software engineering*, pages 55–73. Springer, 2006.

[8] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 553–567, Washington, DC, USA, 2012. IEEE Computer Society.

[9] Marcelo Carlomagno Carlos, Jean Everson Martina, Geraint Price, and Ricardo Felipe Custódio. A proposed framework for analysing security ceremonies. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT 2012)*, pages 440–445, 2012.

[10] Lorrie Faith Cranor. A framework for reasoning about the human in the loop. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*, UPSEC'08, pages 1:1–1:15, Berkeley, CA, USA, 2008. USENIX Association.

[11] Cas Cremers and Sjouke Mauw. *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography. Springer, 2012.

[12] Rachna Dhamija, J. D. Tygar, and Marti A. Hearst. Why phishing works. In Rebecca E. Grinter, Tom Rodden, Paul M. Aoki, Edward Cutrell, Robin Jeffries, and Gary M. Olson, editors, *Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pages 581–590. ACM, 2006.

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

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

[15] Bonnie E. John and David E. Kieras. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, December 1996.

[16] Nikolaos Karapanos, Claudio Marforio, Claudio Soriente, and Srdjan Capkun. Sound-Proof: Usable Two-Factor Authentication Based on Ambient Sound. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 483–498, 2015.

[17] Gavin Lowe. A Hierarchy of Authentication Specifications. In *10th Computer Security Foundations Workshop (CSFW 1997), June 10-12, 1997, Rockport, Massachusetts, USA*, pages 31–44. IEEE Computer Society, 1997.

[18] Mohammad Mannan and Paul C. van Oorschot. Leveraging personal devices for stronger password authentication from untrusted computers. *Journal of Computer Security*, 19(4):703–750, 2011.

[19] Ueli Maurer and Pierre Schmid. A Calculus for Security Bootstrapping in Distributed Systems. *Journal of Computer Security*, 4(1):55–80, 1996.

[20] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In Natasha Sharygina and Helmut Veith, editors, *25th International Conference on Computer Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pages 696–701, Saint Petersburg, Russia, July 2013. Springer.

[21] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[22] Bryan Parno, Cynthia Kuo, and Adrian Perrig. Phoolproof phishing prevention. In *Financial Cryptography and Data Security, 10th International Conference, FC 2006, Revised Selected Papers*, pages 1–19. Springer, 2006.

[23] James Reason. *Human Error*. Cambridge University Press, 1990.

[24] Rimvydas Rukšėnas, Paul Curzon, and Ann Blandford. Modelling and analysing cognitive causes of security breaches. *Innovations in Systems and Software Engineering*, 4(2):143–160, 2008.

[25] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF 2012)*, pages 78–94, 2012.

[26] Sean W. Smith. Humans in the Loop: Human-Computer Interaction and Security. *IEEE Security and Privacy*, 1(3):75–79, May 2003.

[27] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1):191–230, 1999.

APPENDIX

A. Formal Security Properties

We now formally define all the security properties we used. As in *Lowe* [17], we use the actions $\texttt{Running}(A, B, m)$ and $\texttt{Commit}(A, B, m)$ to denote that a non-human agent A has certain beliefs about her apparent communication partner B and about the message m.

Entity authentication of an agent H to another agent S holds if whenever S commits to H, H has taken some action between the start of S and the claim. This corresponds to the *recent aliveness* property of *Cremers and Mauw* [11].

Definition 2. *A protocol model $\mathcal{R}$ satisfies the* entity authentication *property if*

$$\begin{aligned}
&\forall tr \in TR(\mathcal{R}), S, H, m, tr', tr'' : \\
&\qquad tr = tr' \cdot \texttt{Commit}(S, H, m) \cdot tr'' \\
&\Rightarrow \exists tr_1, tr_2, i, l, P, m_2 : tr' = tr_1 \cdot tr_2 \wedge \texttt{Start}(S, i) \in tr_1 \wedge \\
&\quad (\texttt{Send}(H, l, P, m_2) \in tr_2 \vee \texttt{Receive}(H, l, P, m_2) \in tr_2 \\
&\qquad \vee \texttt{Fresh}(H, m_2) \in tr_2 \vee \texttt{Start}(H, m_2) \in tr_2).
\end{aligned}$$

For the special case of authenticating a human H to a server S, we define *device authentication*. Under the assumption that H has exclusive access to a device D and always carries D on him, this property holds if: whenever S commits to H, the device D that belongs to H has performed some action between the start of S and the claim. To denote that the device D belongs to the human H, we introduce the action $\texttt{Device}(H, D)$.

Definition 3. *A protocol model $\mathcal{R}$ satisfies the* device authentication *property if*

$$\begin{aligned}
&\forall tr \in TR(\mathcal{R}), S, H, m, tr', tr'' : \\
&\qquad tr = tr' \cdot \texttt{Commit}(S, H, m) \cdot tr'' \\
&\qquad \Rightarrow \exists tr_1, tr_2, D, i, l, P, m_2 : tr' = tr_1 \cdot tr_2 \wedge \\
&\qquad \texttt{Start}(S, i) \in tr_1 \wedge \texttt{Device}(H, D) \in tr' \wedge \\
&\quad (\texttt{Send}(D, l, P, m_2) \in tr_2 \vee \texttt{Receive}(D, l, P, m_2) \in tr_2 \\
&\qquad \vee \texttt{Fresh}(D, m_2) \in tr_2 \vee \texttt{Start}(D, m_2) \in tr_2).
\end{aligned}$$

We define *message authentication* from a human H to another agent S as: whenever S commits to H and to a message m, then H has previously sent the message m. More specifically, H could have sent m as part of a larger, concatenated message. To denote this, we use the function $\vdash_H$ as introduced in Section IV-C.

Definition 4. *A protocol model $\mathcal{R}$ satisfies the* message authentication *property if*

$$\begin{aligned}\forall tr \in TR(\mathcal{R}), S, H, m, tr', tr'' :\\ tr = tr' \cdot \texttt{Commit}(S, H, m) \cdot tr''\\ \Rightarrow \exists l, P, t', m', t : \texttt{Send}(H, l, P, \langle t', m' \rangle) \in tr'\\ \wedge \langle t', m' \rangle \vdash_H \langle t, m \rangle.\end{aligned}$$

We define a weaker form of *Lowe*'s [17] data agreement. *Weak data agreement* between two non-human agents S and R on a message m holds if: Whenever S commits to a message m, there is an agent R that has previously claimed `Running` with respect to the same m.

Definition 5. *A protocol model $\mathcal{R}$ satisfies the* weak data agreement *property if*

$$\begin{aligned}\forall tr \in TR(\mathcal{R}), S, R', m, tr', tr'' :\\ tr = tr' \cdot \texttt{Commit}(S, R', m) \cdot tr''\\ \Rightarrow \exists R, S' : \texttt{Running}(R, S', m) \in tr'.\end{aligned}$$

Note that even though S can be sure that there is an agent R with which he agrees on the value of m, he need not know who R is. We say agent S knows with whom he has a weak data agreement if the same property holds and the `Running` claim is performed by the agent R to which S commits.

Definition 6. *A protocol model $\mathcal{R}$ satisfies the* weak data agreement *property and agent S knows with whom he has an agreement if*

$$\begin{aligned}\forall tr \in TR(\mathcal{R}), S, R, m, tr', tr'' :\\ tr = tr' \cdot \texttt{Commit}(S, R, m) \cdot tr''\\ \Rightarrow \exists S' : \texttt{Running}(R, S', m) \in tr'.\end{aligned}$$

Finally, for secrecy we define two new actions: The action $\texttt{Secret}(S, m)$ is a claim that denotes that the agent S believes the message m to be secret. The action $\texttt{K}(m)$ denotes that the adversary knows the message m. Secrecy holds if whenever an agent S believes that the message m is secret, the adversary has not learned the message m.

Definition 7. *A protocol model $\mathcal{R}$ satisfies the* secrecy *property if*

$$\forall tr \in TR(\mathcal{R}), S, m : \texttt{Secret}(S, m) \in tr \Rightarrow \texttt{K}(m) \notin tr.$$

B. Remaining Untrained Human Rules

In Section IV-A, we presented the untrained-human rules for generating fresh facts and storing them in the human's knowledge, as well as for sending and receiving facts over insecure channels. We now present the remaining untrained-human rules. Figure 9 depicts the untrained-human rules for sending and receiving facts over channels that are secure (Rules (HR4) and (HR5)), authentic (Rules (HR6) and (HR7)), and confidential (Rules (HR8) and (HR9)). The only differences are the kind of In and Out facts in the premise and conclusion, respectively. All In and Out facts have the same

$$[!\mathsf{HK}(H, \langle t, x \rangle)] \xrightarrow{\texttt{Send}(H, sec, P, \langle t, x \rangle)} [\mathsf{Out}_{sec}(\langle H, P, \langle t, x \rangle \rangle)] \quad \text{(HR4)}$$

$$[\mathsf{In}_{sec}(\langle P, H, \langle t, x \rangle \rangle)] \xrightarrow{\texttt{Receive}(H, sec, P, \langle t, x \rangle)} [!\mathsf{HK}(H, \langle t, x \rangle)] \quad \text{(HR5)}$$

$$[!\mathsf{HK}(H, \langle t, x \rangle)] \xrightarrow{\texttt{Send}(H, auth, P, \langle t, x \rangle)} [\mathsf{Out}_{auth}(\langle H, P, \langle t, x \rangle \rangle)] \quad \text{(HR6)}$$

$$[\mathsf{In}_{auth}(\langle P, H, \langle t, x \rangle \rangle)] \xrightarrow{\texttt{Receive}(H, auth, P, \langle t, x \rangle)} [!\mathsf{HK}(H, \langle t, x \rangle)] \quad \text{(HR7)}$$

$$[!\mathsf{HK}(H, \langle t, x \rangle)] \xrightarrow{\texttt{Send}(H, conf, P, \langle t, x \rangle)} [\mathsf{Out}_{conf}(\langle H, P, \langle t, x \rangle \rangle)] \quad \text{(HR8)}$$

$$[\mathsf{In}_{conf}(\langle P, H, \langle t, x \rangle \rangle)] \xrightarrow{\texttt{Receive}(H, conf, P, \langle t, x \rangle)} [!\mathsf{HK}(H, \langle t, x \rangle)] \quad \text{(HR9)}$$

Fig. 9: Untrained-human rules for sending messages to and receiving messages from secure, authentic, and confidential channels.

structure: The first argument A denotes the sender, the second argument B the receiver, and the third one the message m with its tag t.

C. Tags

We formally define how tags are assigned to the terms in the protocol specification. We only need the tags to denote the human's interpretation of messages. A human can only concatenate messages and split concatenated messages. Therefore, we define a tagging function that gives an interpretation to the messages that are part of a larger concatenated term.

First, we define a function $t(m)$ that assigns a tag to every atomic message m. That is, every public and fresh constant as well as every variable and function symbol is assigned a unique tag. We then inductively define tags for composed messages as follows. Whenever a message consists of a function f applied to arguments $a_1, ..., a_n$, the message's tag consists of the tag for the function symbol f concatenated with the tags of $a_1, ..., a_n$. We then map the resulting concatenated tags to public constants. Finally, we define the function $Tag(m)$. If the top level function of a message m consists of a pairing function with arguments m_1 and m_2, then the final tag of m is a pair consisting of the tags of m_1 and m_2. Whenever the top level function is not a pair, the message is tagged with the public constant as previously defined.

Formally, we first choose an injective function $t(m) : \Sigma \cup X \to \mathcal{L}$ that assigns a unique term from the countably infinite

set of tags $\mathcal{L}$ to every symbol in Σ and X. Then we inductively define the injective function $T(m) : \mathcal{T}_\Sigma(X) \to \mathcal{L}^*$ as follows.

$$T(m) = \begin{cases} t(m) & \text{if } m \in \Sigma \cup X \\ t(f) \cdot T(a_1) \cdot ... \cdot T(a_n) & \text{if } m = f(a_1, ..., a_n), \\ & f \in F_{sym}, a_i \in \mathcal{T}_\Sigma(X). \end{cases}$$

This function assigns to every term m a sequence of tags. Since $\mathcal{L}^*$ and $\mathcal{C}_{pub}$ are countably infinite sets, there is an injective function $i : \mathcal{L}^* \to \mathcal{C}_{pub}$. We define $preTag(m) : \mathcal{T}_\Sigma(X) \to \mathcal{C}_{pub}$ as a composition of the functions i and T. Finally, we construct an injective function

$$Tag(m) : \mathcal{T}_\Sigma(X) \to \mathcal{T}_{\mathcal{C}_{pub} \cup \{\text{pair}\}}(X)$$

such that $Tag(m)$ satisfies the equation $Tag(\langle m_1, m_2 \rangle) = \langle Tag(m_1), Tag(m_2) \rangle$. The construction is given by

$$Tag(m) = \begin{cases} \langle Tag(m_1), Tag(m_2) \rangle & \text{if } m = \langle m_1, m_2 \rangle \\ preTag(m) & \text{otherwise.} \end{cases}$$

D. MP-Auth_MA with Rule-Based Humans

We give a precise statement and proof for the fact that *MP-Auth_MA* does not provide message authentication from a human to a server for any rule-based human with the given guidelines in our model.

The purpose of the *MP-Auth_MA* protocol is the authenticated transmission of a message from the human H to the server S. The associated security property, message authentication, is formally stated in Definition 4. This property is trivially satisfied if S never performs a commit claim. A guideline that prevents the execution of the protocol leads therefore to a trivially satisfied authentication claim. Such a guideline is not useful, however, since it does not achieve the protocol's purpose. We are therefore only interested in guidelines that allow the human to execute the protocol in a manner that the protocol's purpose is achieved. Formally, for every security property that is universally quantified over the set of all protocol traces we define a corresponding functionality property that is existentially quantified over the set of all protocol traces. For example, the functionality property of a message authentication protocol is satisfied if there exists a trace that reaches the commit claim and satisfies message authentication. We then say that a protocol *non-vacuously satisfies* a security property, if it satisfies both the security property and the corresponding functionality property.

We can now state the claim that our exemplary set of guidelines is insufficient for *MP-Auth_MA* to satisfy message authentication from the human to the server.

Claim 13. *There is no combination of the $NoTell$, $NoTellExcept$, $NoGet$, and $ICompare$ guidelines for the rule-based human such that* MP-Auth_MA *non-vacuously satisfies message authentication from the human to the server.*

Proof. We show that for all combinations of our exemplary predicates, the protocol does not satisfy the message authentication property or does not satisfy the corresponding functionality property.

Let *'Human'* be a rule-based human that behaves according to a combination of the exemplary guidelines.

Consider the trace AT shown in Figure 10. The trace is valid if the *MP-Auth_MA* protocol is executed with the untrained human agent. The trace does not satisfy the message authentication property stated in Definition 4, because it contains a `Commit` action, but no corresponding `Send` action. By Claims 14, 15, 17 and 19 below, AT satisfies all combinations of predicates that do not include $NoTell(\textit{'Human'}, \textit{'OK'})$ and $NoTellExcept(\textit{'Human'}, \textit{'OK'}, E)$, for $E \in \{\textit{'OK'}, \textit{'P'}, \textit{'S'}, \textit{'m'}, \textit{'idH'}, \textit{'idS'}\}$. Therefore, for all these combinations, the protocol does not satisfy the message authentication property.

It remains to consider combinations of predicates that include $NoTell(\textit{'Human'}, \textit{'OK'})$ or $NoTellExcept(\textit{'Human'}, \textit{'OK'}, E)$ for some $E \in \{\textit{'OK'}, \textit{'P'}, \textit{'S'}, \textit{'m'}, \textit{'idH'}, \textit{'idS'}\}$. We show for each of these predicates that the protocol violates the message authentication functionality property.

By Definition 4, a necessary requirement for the *MP-Auth_MA* protocol to non-vacuously satisfy the message authentication property is that the server reaches the specified `Commit` action. For simplicity, we call this the $Functional$ predicate:

$$Functional(S, H, m, tr) := \exists \texttt{Commit}(S, H, m) \in tr.$$

By Claims 16 and 18, $NoTell(\textit{'Human'}, \textit{'OK'})$ and $NoTellExcept(\textit{'Human'}, \textit{'OK'}, E)$ for all $E \in \{\textit{'OK'}, \textit{'P'}, \textit{'S'}, \textit{'m'}, \textit{'idH'}, \textit{'idS'}\}$ contradict the $Functional$ predicate. Thus the *MP-Auth_MA* protocol does not non-vacuously satisfy the message authentication property for any combination of our examplary predicates. □

The following claims are used in the proof of the preceding claim.

Claim 14. *The attack trace AT satisfies the predicate $NoGet(\textit{'Human'}, x)$ and the predicate $ICompare(\textit{'Human'}, x)$ for any tag x.*

Proof. As there are no `Receive` events of the human *'Human'* in the attack trace, the predicates are satisfied for any tag x. □

Consequently, $NoGet$ and $ICompare$ do not avoid the attack trace AT. Next, we consider $NoTell$ and $NoTellExcept$. Because there is a $\texttt{Send}(\textit{'Human'}, sec, D, \langle \textit{'OK'}, OK \rangle)$ event in AT, we distinguish between the case where the tag is equal to *'OK'* and the case where it is not. We start with the latter.

Claim 15. *The attack trace AT satisfies the predicate $NoTell(\textit{'Human'}, x)$ and the predicate $NoTellExcept(\textit{'Human'}, x, E)$ for any tag E and for any tag x that is not equal to 'OK'.*

Proof. As there are no `Send` events of the human *'Human'* for a tag x which is unequal to *'OK'*, the predicates are satisfied for any E and any x unequal to *'OK'*. □

$$
\begin{aligned}
AT = [&\{\texttt{InitK}(\textit{'Human'}, \langle \textit{'OK'}, OK\rangle),\\
&\texttt{InitK}(\textit{'Human'}, \langle \textit{'D'}, D\rangle),\\
&\texttt{InitK}(\textit{'Human'}, \langle \textit{'P'}, P\rangle),\\
&\texttt{InitK}(\textit{'Human'}, \langle \textit{'S'}, S\rangle),\\
&\texttt{InitK}(\textit{'Human'}, \langle \textit{'m'}, mH\rangle),\\
&\texttt{InitK}(\textit{'Human'}, \langle \textit{'idH'}, idH\rangle),\\
&\texttt{InitK}(\textit{'Human'}, \langle \textit{'idS'}, idS\rangle)\}^b,\\
&\{\texttt{Start}(S, \langle skS, \textit{'Human'}, idS, idH, rS, rD\rangle)\}^b,\\
&\{\texttt{Start}(D, \langle \textit{'Human'}, idH, pk(skS), idS, S, rS, rD\rangle)\}^b,\\
&\{\texttt{Receive}(S, \textit{ins}, P, m)\}^b,\\
&\{\texttt{Fresh}(S, rS_2)\}^b,\\
&\{\texttt{Send}(S, \textit{ins}, P, \mathrm{senc}(\langle m, rS_2\rangle, f(rS, rD)))\}^b,\\
&\{\texttt{Receive}(D, \textit{ins}, P, \mathrm{senc}(\langle m, rS_2\rangle, f(rS, rD)))\}^b,\\
&\{\texttt{Send}(D, \textit{sec}, \textit{'Human'}, \langle \textit{'m'}, m\rangle)\}^b,\\
&\{\texttt{Send}(\textit{'Human'}, \textit{sec}, D, \langle \textit{'OK'}, OK\rangle)\}^b,\\
&\{\texttt{Receive}(D, \textit{sec}, \textit{'Human'}, \langle \textit{'OK'}, OK\rangle)\}^b,\\
&\{\texttt{Send}(D, \textit{ins}, P, \mathrm{senc}(f(m, rS_2), f(rS, rD)))\}^b,\\
&\{\texttt{Receive}(S, \textit{ins}, P, \mathrm{senc}(f(m, rS_2), f(rS, rD)))\}^b,\\
&\{\texttt{Commit}(S, \textit{'Human'}, m)\}^b]
\end{aligned}
$$

Fig. 10: A trace of *MP-Auth_MA* violating message authentication. Multisets are denoted by $\{\}^b$.

We now consider the special case where the tag is equal to *'OK'*. We establish with Tamarin:

Claim 16. *$NoTell$('Human', 'OK') contradicts the $Functional$ predicate of the protocol.*

For this reason, *$NoTell$('Human', 'OK')* is not applicable. It remains to show what happens with *$NoTellExcept$('Human', 'OK', E)*. We first distinguish the case where E corresponds to the tag that *'Human'* associates with his device.

Claim 17. *The attack trace AT is not excluded by the predicate $NoTellExcept$('Human', 'OK', 'D') where 'D' is the tag that the human associates with his device in the initial knowledge.*

Proof. The attack trace AT satisfies the trace because the human only sends *'OK'* to the device on a secure channel. □

Next, we analyze the case where E corresponds to any other tag that appears in the initial knowledge of *'Human'*. We establish with Tamarin:

Claim 18. *$NoTellExcept$('Human', 'OK', E) with the tag $E \in$ {'OK','P','S','m','idH','idS'}, contradicts the $Functional$ predicate of the protocol.*

Finally, we consider the case where E corresponds to any other tag that is not appearing in the initial knowledge of *'Human'*.

0. H : $\mathrm{knows}(D, P, S, pw, idH)$
0. D : $\mathrm{knows}(H, k_{DS})$
0. S : $\mathrm{knows}(H, D, pw, idH, k_{DS})$
1. H $\bullet\!\!\rightarrow\!\!\bullet$ P : S, idH
2. P $\circ\!\!\rightarrow\!\!\bullet$ S : idH
3. S $\bullet\!\!\rightarrow\!\!\bullet$ P : $\mathrm{fresh}(r).\mathrm{senc}(r, k_{DS})$
4. P $\bullet\!\!\rightarrow\!\!\bullet$ D : $\mathrm{senc}(r, k_{DS})$
5. D $\bullet\!\!\rightarrow\!\!\bullet$ H : $otpw(r)$
6. H $\bullet\!\!\rightarrow\!\!\bullet$ P : $pw, otpw(r)$
7. P $\bullet\!\!\rightarrow\!\!\bullet$ S : $pw, otpw(r)$

Fig. 11: Protocol *Cronto*

Claim 19. *The attack trace AT is not excluded by the predicate $NoTellExcept$('Human', 'OK', E) with a tag $E \notin$ {'OK','D','P','S','m','idH','idS'}.*

Proof. The implication in the definition of $NoTellExcept$ is alway true if its third argument is not a tag that appears in the human's initial knowledge. Therefore, for $E \notin$ {'OK','D','P','S','m','idH','idS'}, all traces are allowed and the attack trace AT is not excluded. □

E. Case Study Protocols

We now provide more details on the protocols used in Section VI. As already stated, in all the protocols a human H wants to authenticate to a server S. For this purpose he has access to a platform P and exclusive access to a personal device D. Moreover, idH is always the human's identity, pw a password, and $otpw$ a one-time password that is freshly generated in each run. Recall, that when not stated otherwise, we assume the channels between H and P as well as the channels between H and D to be secure. Further, we model a TLS connection between P and S by assuming that the first message from P to S is sent on a confidential channel and the subsequent messages on a secure channel.

A variation of the protocol *Cronto* can be used for transaction authentication. We call this version *Cronto_MA* because we examine if it satisfies the message authentication property. In all the other protocols, depicted in Figures 13–16, m denotes the message that we have explicitly added to examine message authentication (see Section VI-B2). This means that in the original versions of the protocols m is empty.

All the protocols except for *Phoolproof* start with the human entering his identity and the name of the server he wants to contact on the platform P. Afterward, P relays the identity of H to the server S.

1) Cronto: In this protocol, shown in Figure 11, the server shares a secret key k_{DS} with the device. After having received the human's identity, the server S generates a fresh nonce r and encrypts it with k_{DS} in the form of a cryptogram. S then sends the cryptogram to P where it is displayed for the human. The human uses the device D to scan the cryptogram. We assume that the scanning of the graphic is done over a

0. H : $\text{knows}(D, P, S, m, idH)$
0. D : $\text{knows}(H, k_{DS})$
0. S : $\text{knows}(H, D, idH, k_{DS})$
1. H $\bullet\!\!\rightarrow\!\!\bullet$ P : idH, S, m
2. P $\circ\!\!\rightarrow\!\!\bullet$ S : idH, m
3. S $\bullet\!\!\rightarrow\!\!\bullet$ P : $\text{fresh}(r).\text{senc}(\langle r, m\rangle, k_{DS})$
4. P $\bullet\!\!\rightarrow\!\!\bullet$ D : $\text{senc}(\langle r, m\rangle, k_{DS})$
5. D $\bullet\!\!\rightarrow\!\!\bullet$ H : $m, otpw(r, m)$
6. H $\bullet\!\!\rightarrow\!\!\bullet$ P : $otpw(r, m)$
7. P $\bullet\!\!\rightarrow\!\!\bullet$ S : $otpw(r, m)$

Fig. 12: Protocol *Cronto_MA*

0. H : $\text{knows}(P, D, S, pw, m, idH)$
0. D : $\text{knows}(H)$
0. S : $\text{knows}(H, pw, D, idH)$
1. H $\bullet\!\!\rightarrow\!\!\bullet$ P : S, idH, pw, m
2. P $\circ\!\!\rightarrow\!\!\bullet$ S : idH, m
3. S $\circ\!\!\rightarrow\!\!\bullet$ D : $\text{fresh}(c).c, m$
4. D $\bullet\!\!\rightarrow\!\!\bullet$ H : c, m
5. H $\bullet\!\!\rightarrow\!\!\bullet$ P : S, c
6. P $\bullet\!\!\rightarrow\!\!\bullet$ S : c, pw, m

Fig. 13: Protocol *Google 2-Step*

secure channel from P to D. D decrypts the cryptogram with the known key and computes a one-time password from the retrieved r that it displays to the human. H then enters the one-time password together with his password on P, which relays it to S.

Cronto_MA, depicted in Figure 12, denotes the version of the protocol that can be used for transaction authentication. The human H first enters his identity idH, the server S he wants to contact, and a message m on the platform P. Afterward, P relays idH and m to the server S and S computes a fresh nonce r. In contrast to *Cronto* for entity authentication, Figure 11, S computes the cryptogram over both the nonce r and the message m. The result is again sent to P and scanned with the device D, which decrypts it. The one-time password that D computes next also contains the nonce r and the message m that H sent. D displays the one-time password together with m to H. H checks if it is the right m and if it is he enters $otpw$ on P which relays it to S.

2) Google 2-Step: Figure 13 depicts Google's two-factor authentication. Upon being contacted by the platform P of the human H, the server S generates a fresh code c. The code is then sent to the device D, for example by SMS, i.e., by text message. We assume a confidential SMS channel. In the final steps, the human reads the code from the device, enters it on P, and P sends the code and the password to the server.

3) OTP over SMS: Figure 14 models the protocol *OTP over SMS* in which the server S sends a one-time password to the human's device D. *Bonneau et al.* [8] do not describe a specific protocol that they have in mind but rather say that one-time passwords can be used in different ways. Among others, it is pointed out that they can be used as a second factor.

0. H : $\text{knows}(D, P, S, m, idH)$
0. D : $\text{knows}(H)$
0. S : $\text{knows}(H, D, idH)$
1. H $\bullet\!\!\rightarrow\!\!\bullet$ P : S, idH, m
2. P $\circ\!\!\rightarrow\!\!\bullet$ S : idH, m
3. S $\circ\!\!\rightarrow\!\!\bullet$ D : $\text{fresh}(otpw).otpw, m$
4. D $\bullet\!\!\rightarrow\!\!\bullet$ H : $otpw, m$
5. H $\bullet\!\!\rightarrow\!\!\bullet$ P : $otpw$
6. P $\bullet\!\!\rightarrow\!\!\bullet$ S : $m, otpw$

Fig. 14: Protocol *OTP over SMS*

0. H : $\text{knows}(D, P, S, pw, m, idH)$
0. D : $\text{knows}(H, S, skD, idH, \text{pk}(skS))$
0. S : $\text{knows}(H, D, idH, pw, skS, \text{pk}(skD))$
1. H $\bullet\!\!\rightarrow\!\!\bullet$ D : S, m
2. D $\bullet\!\!\rightarrow\!\!\bullet$ P : S
3. P $\circ\!\!\rightarrow\!\!\circ$ S : *'hello'*
4. S $\circ\!\!\rightarrow\!\!\circ$ P : $\text{fresh}(chall).chall, \text{sign}(chall, skS)$
5. P $\bullet\!\!\rightarrow\!\!\bullet$ D : $chall, \text{sign}(chall, skS)$
6. D $\bullet\!\!\rightarrow\!\!\bullet$ P : $idH, m, \text{sign}(\langle idH, chall, m\rangle, skD)$
7. P $\circ\!\!\rightarrow\!\!\bullet$ S : $idH, m, \text{sign}(\langle idH, chall, m\rangle, skD)$
8. H $\bullet\!\!\rightarrow\!\!\bullet$ P : pw, idH
9. P $\circ\!\!\rightarrow\!\!\bullet$ S : idH, m, pw

Fig. 15: Protocol *Phoolproof*

If we use a one-time password as a second factor, together with a normal password, the protocol closely resembles the *Google 2-Step* protocol. For this reason, we chose to model a protocol that uses a one-time password as a single factor. We model the one-time password by a random nonce $otpw$ that the server S freshly generates in each protocol run. $otpw$ is sent to the device D over an SMS channel, which we assume to be confidential. $otpw$ is then displayed on the device for the human. Finally, the human enters $otpw$ on the platform, from where it is sent to the server.

4) Phoolproof: Figure 15 depicts *Phoolproof*, the only protocol we examine that does not start with the human telling his platform to which server he wants to connect. Instead, the human selects on his device D the server S from a list of registered servers. The device then communicates this choice to the platform P, over a Bluetooth channel, which we assume to be secure. Next, the platform P sends a 'hello' message to the sever S. We assume that the communication between P and S is first insecure and then, from Step 7 onwards, confidential from P to S, which models the start of the TLS connection. Upon receiving the initialization message from P, S sends a signed challenge $chall$ over P to D. D then signs the identity of the human idH and the challenge $chall$ and sends idH, together with the signature, back to S, again via P. Finally, the human enters his password and idH on P and P relays it to the server.

5) Sound-Proof: Recall that the idea of *Sound-Proof*, shown in Figure 16, is to authenticate a human H who logs into platform P by measuring if his device D is in the proximity

0. H : $\mathrm{knows}(pw, P, D, S, idH)$
0. D : $\mathrm{knows}(H, S, skD, OK)$
0. S : $\mathrm{knows}(H, D, \mathrm{pk}(skD), pw, idH)$
1. H $\bullet\!\!\rightarrow\!\!\bullet$ P : S, idH, pw
2. P $\circ\!\!\rightarrow\!\!\bullet$ S : idH, pw
3. S $\bullet\!\!\rightarrow\!\!\bullet$ P : *'record'*, $\mathrm{pk}(skD)$
4. S $\bullet\!\!\rightarrow\!\!\bullet$ D : *'record'*
5. P, D : $\mathrm{fresh}(r)$
6. P $\bullet\!\!\rightarrow\!\!\bullet$ S : $\mathrm{fresh}(k).\mathrm{senc}(\langle S, r\rangle, k)$, $\mathrm{aenc}(k, \mathrm{pk}(skD))$
7. S $\bullet\!\!\rightarrow\!\!\bullet$ D : $\mathrm{senc}(\langle S, r\rangle, k), \mathrm{aenc}(k, \mathrm{pk}(skD))$
8. D $\bullet\!\!\rightarrow\!\!\bullet$ S : OK

Fig. 16: Protocol *Sound-Proof*

of P. The measurement of whether D and P are at the same location is done by measuring ambient noise. In a setup phase prior to protocol execution, the human H registers his device. For this reason, S knows which device D belongs to H and also knows D's public key. In the setup phase, the application on D is also bound to the user's account on the server S. We therefore assume that D and S know each other and model the TLS connection between S and D with secure channels. After the server S receives from P the identity and the password from the user, S sends a *'record'*-command to both the device and the platform. S also sends the public key of D to P. It is assumed that no one can be in the same location as P and D. We therefore model the ambient noise by a fresh nonce that is only given to the platform and the device. P then encrypts the measured information r and S with a fresh symmetric key k. Further, P encrypts k with the public key of D. The encrypted messages are sent, over the server S, to D. Finally, D decrypts the measurement r and compares it to the measurement it previously made. In the actual protocol, D accepts and sends an OK message to the server if the measurements are roughly the same. In our model, we require that D receives the same nonce that it previously stored. An advantage of the protocol is that it reduces the overhead for the human of having to interact with the device.