Probabilistic Modelling of Humans in Security Ceremonies

Christian Johansen(✉) and Audun Jøsang

Department of Informatics, University of Oslo,
Blindern, P.O. Box 1080, 0316 Oslo, Norway
{cristi,audun}@ifi.uio.no

Abstract. We are interested in formal modelling and verification of security ceremonies. Considerable efforts have been put into verifying security protocols, with quite successful tools currently being widely used. The relatively recent concept of security ceremonies, introduced by Carl Ellison, increases the complexity of protocol analysis in several directions: a ceremony should include all relevant out-of-bad assumptions, should compose protocols, and should include the human agent. Work on modelling human agents as part of IT systems is quite limited, and the few existing studies come from psychology or sociology. A step towards understanding how to model and analyse security ceremonies is to integrate a model of human agents with models for protocols (or combination of protocols). Current works essentially model human agent interaction with a user interface as a nondeterministic process.

In this paper we propose a more realistic model which includes more information about the user interaction, obtained by sociologists usually through experiments and observation, and model the actions of a human agent as a probabilistic process. An important point that we make in this paper is to separate the model of the human and the model of the user interface, and to provide a "compilation" operation putting the two together and encoding the interaction between the human and the interface. We base our work on a recently proposed model for security ceremonies, which we call the Bella-Coles-Kemp model.

1 Introduction

The motivation for analysing security ceremonies is well articulated in [10,31] with convincing examples. Technically, security ceremonies are meant to extend security protocols by including human agents in the formalization, and by explicitly including aspects of the environment and potential attackers. A ceremony may also combine protocols. In consequence, a formalism for security ceremonies is expected to be expressive enough to include existing formalisms for protocols as special cases. Such a formalism should offer the possibility to model human behavior related to the ceremony.

This work was partially supported by the project OffPAD with number E!8324 part of the Eurostars program funded by the EUREKA and European Community.

With the risk of appearing pedantic to some readers, we give some motivation for using formal techniques for security protocol analysis. Arguably, for security systems perfection and assurance of perfection are highly important, since bugs cannot be considered "features". For a security system one often wants to be provided with guarantees that some expected security properties are met. This can be even more difficult to achieve for security ceremonies, which are more complex, composing protocols, including hidden assumptions, and human agent models. In case of security protocols one can hardly rely on testing to provide assurance; and experience has shown that protocols that are thoroughly tested in practice for years turn out to contain serious flaws, where an infamous example is [21].

The need to eliminate hidden flaws in security protocols is the motivation for developing mathematical models and theories for studying security protocols. The practical results of such efforts are the formal tools based on the underlying mathematical theories. These tools offer the ability to have a (semi-)automated way of ensuring security properties. Examples of tools include model checkers like Murphi [22,38] or FDR [13] which are push-button tools with yes/no answers; or theorem provers, which often need interaction with expect users but which achieve stronger results than model checkers do, with examples like ProVerif [6] for the symbolic (process calculi) approach, CryptoVerif [7] for the computational approach, or Isabelle/HOL [25].

We are interested in formal models for security ceremonies, and our work is inspired by the rather limited, recent literature on this topic [8,10,26,27,29,31]. Because of the complexity of the problem, this paper focuses only on the aspect of integrating the human agent into the ceremony, together with a user interface. Therefore, in this paper we concentrate on the human-computer interaction, which forms the layer III in the Bella-Coles-Kemp model [3] for security ceremonies. This is a general/abstract model introduced with similar purposes as the Dolev-Yao model. In fact, the Dolev-Yao model [9] could form the first layer of the Bella-Coles-Kemp model, i.e., the layer of the network communication protocols. But for ceremonies, many other factors need to be considered, including social factors, and this is what the Bella-Coles-Kemp model tries to identify. This model has also been used by [11]. One of our purposes is to extend the work of [3], where layer III was modeled as a nondeterministic process, to the more general probabilistic setting here.

A security ceremony involves a user interface to collect any needed input, like passwords, from the human participant in the ceremony. Generally speaking, several different user interfaces can be available to the same human user; and there can be user interfaces for several of the components of the ceremony, like for the different protocols involved. But for simplicity we restrict our discussion for now to a single user interface, which is supposed to provide the required input information to a security protocol.

The few studies on modelling human interacting with a user interface using formal methods [3,33] are based on a nondeterministic approach. Our intention in this paper is to extend these approaches to a *probabilistic* model. Moreover,

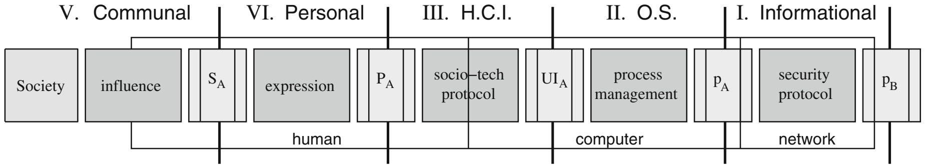


Fig. 1. The Bella-Coles-Kemp model (BCK model) for security ceremonies. (Picture taken from [3, Fig. 1])

these approaches usually provide one model that captures the total human-computer interaction. We split this into: one probabilistic model for the human, using the notion of "persona" from user-centred design, one model for the user interface, and a notion of "compilation" which puts the two models together and captures the interactions between the human and the user interface.

The structure of this paper is as follows: we explain the Bella-Coles-Kemp model in Sect. 2, introduce probabilistic models of humans and their interaction with the user interface in Sect. 3, and investigate ways of putting together the human model with the UI model, which we call "compilation" in Sect. 4.

2 The Bella-Coles-Kemp Model

We refer to the model introduced in [3] as "the Bella-Coles-Kemp model" and sometimes use the abbreviation BCK model. This is a rather abstract model for security ceremonies, which provides a good common basis for formal analysis by offering a framework of reference for defining models, attackers, and reasoning tools. The BCK model, pictured in Fig. 1, extends and includes standard models for analysing security protocols, like the Dolev-Yao model [9]. The standard security protocols would form only the layer I of the BCK model.

When explaining the BCK model it is good to make correlations with the existing approaches for analysing security protocols, usually based on the Dolev-Yao assumptions, and using specification languages like the spi calculus [2] or applied pi-calculus [1]. Usually, security protocols are formed of the parties (or players) and the interaction medium they use for communication (or any other exchange of information). The parties are usually honest, whereas the intruder (attacker) controls the interaction medium. More than two parties can be involved in the protocol, but the standard protocols consist of two honest parties, Alice and Bob. Third parties, usually dishonest, appear due to the ability of the intruder to disguise as a party in any number of protocol runs. The Dolev-Yao model defines the powers that the attacker has over the interaction medium, like power to delete, change, or insert messages, to and from any other party. As there exist many kinds of security protocols (e.g., with multiple honest parties, with different attacker possibilities, etc.) we will confine our presentation here to the basic definitions which can be carried over to the BCK model.

In the BCK model the parties form the light boxes, whereas the interaction medium forms the dark boxes. The parties appear at different layers of the BCK model and in different abstractions; i.e., the light boxes represent the players in the respective layer, which are abstractions of the parties or are controlled by the parties. In the layer I (also called "Informational") we encounter the processes p_A and p_B controlled by Alice respectively Bob, which are running the computers of Alice and Bob, communicating through the network, i.e., the dark box. This layer I would thus be subject to the standard Dolev-Yao assumptions.

In BCK other players appear at the other layers: at layer II (also called "Operating System") *the user interface* UI_A associated to Alice, which interacts with the computer process p_A, e.g., by sending information taken from the user required by the security protocol run by p_A, like a password or biometrics. The same UI_A interacts at layer III (also called "Human-Computer Interaction") with *a "persona"* P_A of Alice for this particular ceremony. The persona has interaction at layer IV (also called "Personal") with *the self* S_A of Alice, which in turn is influenced by the Society through various social protocols at layer V.

Players may interact only as part of a layer, and one layer may involve players pertaining to different users. Important to note is that in BCK one player usually is involved in two (adjacent) layers.

Research in computer science until now has mostly focused on layer I, and largely ignored layers II–V. We see layer II as pertaining also to the technological community, the same as layer I has been until now; whereas layer V would pertain to the sociology community. Layer IV would be investigated more by the psychology researchers. Layer III on the other hand is at the interaction between technological and social sciences, with a rapidly evolving field, having terminology s.a.: HCI, user-centred design [4], interaction design [32], etc.

The usefulness of the BCK model is also to make obvious the need of collaboration between these fields of social, psychology, and technology, in order to tackle the complexities of the security ceremonies. One can very well focus on individual layers, but the BCK model brings the isolated results into the general picture which eventually needs to be handled in order to claim security results of practical use.

The BCK model is abstract and general, and we see works in the future detailing on all the new layers II–V, the same as has been done until now with the layer I. The interaction medium, the dark box, can be split into more fine-grained divisions, and each division would have its protocol and assumptions. For layer II it is more easy for computer scientists to bring their knowledge of operating systems design and see that a UI could consist of a screen and its driver, a display client like a browser displaying an input form, a keyboard with its drivers, and the many other components that transport the information between these many UI components and the end process p_A. The same in the social protocols of layer V, where various means of social manipulation exist, and quantitative and qualitative measures could be devised for analysing their usefulness in terms of power to influence, e.g., depending on the social scale or training level of the users, i.e., the self S_A.

In this paper we take the stand that layer III has not so many division possibilities, and think of the possible interactions in the dark box as rather simplistic, like matching of actions on the persona and UI sides. But various kinds of interactions can be thought of, and we capture them with a notion of *compilation*. Various compilation operations can be devised for various kinds of simple interactions between personas and UIs, and we detail on these in Sect. 4. Compilation would put together the model for the persona and the model for the UI, to form a single model for the whole layer III, which would include the two players and the dark box. This model would be representing a new player which interacts in the layer II and layer IV.

Attacks can be thought at all the layers of the BCK model [11]:

Layer I attacks: The classic attacks like snooping, brute-force, or involving any of the Dolev-Yao assumptions.

Layer II attacks: The bank account number is typed and seen at UI_A by Alice but is not the same as the one used by p_A in the security interaction with the Bank process p_B.

Layer III attacks: The expressed interaction of P_A is not properly captured by UI_A, e.g.:
- User types digit 0 but the UI discards it;
- User types digit 8 but the UI receives and displays digit 3.

Such attacks can be observed by the persona or not, and detecting them and quantifying the observational possibilities is a problem for models and techniques pertaining to analyses of layer III.

Layer IV attacks: The user is tricked into being *trusting instead of cautious*; which can be done by the attacker through clever use of the colours, logos, security symbols, etc.

3 Probabilistic Modelling of Humans

Some formal models have been proposed for analysing human agents and their behavior when interacting with a user interface. We are inspired by work in the cognitive sciences [23,33] which has also been used to analyse security breaches in [34]; as well as by work from the social sciences [3,20], with a good reference being [36]. We adopt the notion of *persona* to characterize a user in a socio-technical interaction situation. We define a persona to be a set of social and cognitive attributes of a human, including emotions, senses, or memory.

Definition 3.1 (Persona). *An* attribute *is represented as a predicate in some logic (usually propositional logic or first-order logic). A* persona *is defined as a finite set of attributes.*

This definition of persona is related to the one in [36, Lemma 2] in the sense that the attribute predicates can be represented as a computer data structure and thus stored in the digital world (whereas the actual user resides in the physical world). In [36] a persona is a representation of a user from the physical world into the digital world.

We will mostly use attributes as propositional predicates, like *cautious* or *confident*, which are either true or false. But more complex attributes, like memory, may require more expressive logics. It is interesting and certainly useful to investigate deeper the kinds of logics needed to express various persona attributes, but this is outside the scope of this paper, and the work of Blandford et al. [34], which uses first-order logic to model the human involvement in a security protocol, is a good start for such investigations. Also useful would be works on applying first-order dynamic logic to model memory in computer programs, e.g., [15, Part III].

For the layer III that we are concerned with here, a persona is an abstract representation in the digital world of a physical user. Personas represent the user (usually class of users with the same expected qualities) in different particular situations (in our case, w.r.t. the UI), e.g. the user as a: citizen (when accessing her social services) or an employee (when accessing corporate networks).

A user may exhibit different personas at different points in time, and may change from one persona to another depending on the interaction with the user interface. For the same user, we may be interested in different personas for different aspects of the user interface.

Therefore, we will be working with *personas models*, which may contain several personas connected by actions. The actions are the way we model the *change factor*, i.e., that which makes the human exhibit a different persona. Without a change factor there would be no change in the persona. An action can be thought of many things, from passage of time, to interaction with the user interface, to social change coming from the social context, i.e., from the outside of the human-computer interaction system. Examples of actions can be high-level s.a.: "fill-in-form", "provide-explanation", or "make-query"; or concrete s.a.: "press-submit-button", "type-password", "abort". A formalism for actions should be used, like algebras [14] or logics [28,35], but we do not go into this detail here, and keep our presentation simple. Introducing such an action formalism would complicate the models.

Definition 3.2 (Ideal Personas Model). *Consider a countable set of actions Λ and a set of attributes Φ. A* personas model *is a tuple $PM = (Q, T, \Lambda, \Phi)$ with:*

- *$Q \subseteq 2^{\Phi}$ a finite set of personas, and*
- *$T \subseteq Q \times \Lambda \times Q$ a transition relation labeled by actions between personas, with the restriction that T is a partial function between $Q \times \Lambda \rightarrow Q$ (meaning that for each action it is determined how it changes a persona).*

Notation 3.3. *The following notation will be useful. In a personas model $PM = (Q, T, \Lambda, \Phi)$, for some set of transitions $T' \subseteq T$, define $Q[T'] \overset{def}{=} \{q \in Q \mid \exists(q, \alpha, q') \in T' \text{ or } \exists(q', \alpha, q) \in T'\}$; i.e., the set of those personas entering some transition from T'. For some set Λ (of actions usually), we denote by $\Lambda_* \overset{def}{=} \Lambda \cup \{*\}$, where $* \notin \Lambda$ is a special symbol not part of any action set. We sometimes denote transitions from T as $q \xrightarrow{\alpha} q'$.*

A minimal non-trivial personas model has no transitions and one single persona. Other simple personas models can be defined with one single persona and one transition to itself for each action.

Personas models would usually be studied in general settings by sociologists and psychologists, and include many and various actions. But for a specific user interface only a subset of these actions is of interests. Therefore, restrictions of these general personas models would be needed before compiling with the user interface. One way to define such restrictions is as follows, using what we call an *action restriction operation.*

Definition 3.4 (Action Restriction Operation). *For a personas model* $PM = (Q, T, \Lambda, \Phi)$ *and a set of actions* Λ'*, an* action restriction operation *would take an actions map* $f : \Lambda \to \Lambda'_*$*, and return a new personas model* $PM|_{\Lambda'} = (Q|_{\Lambda'}, T|_{\Lambda'}, \Lambda', \Phi)$ *with:*

- $T|_{\Lambda'} \overset{def}{=} \{(q, f(\alpha), q') \mid (q, \alpha, q') \in T \textit{ and } f(\alpha) \neq *\}$,
- $Q|_{\Lambda'} \overset{def}{=} Q[T|_{\Lambda'}]$.

A simple instance of the action restriction operation is when the action map f is a partial identity function that maps any action from $\Lambda \cap \Lambda'$ into itself and any other into $*$. The restricted personas model would thus keep only those transitions that are labelled by actions from $\Lambda \cap \Lambda'$ and discards the rest of the model. But the above definition is open to more possibilities, like when the personas model works with more fine-grained actions, which in the user interface model would be collapsed (i.e., through a specific definition of f) into a single, more abstract, action.

The personas model from above is *deterministic*, meaning that it is completely known how an action changes a persona. This is also why we called it "ideal". But in reality this is not the case because we never have complete information. We are unsure of how an action may change a persona, but empirical studies do give some information. We introduce probabilities to capture the existing knowledge in a more meaningful way. Probabilities allow the model to carry more knowledge, usually accumulated from social studies. Our model extends the existing non-deterministic models in which it is assumed that there is no knowledge about how an action changes a persona (except for the fact that the new persona is part of some restricted subset of possible personas).

Definition 3.5 (Probabilistic Personas Model). *A* probabilistic personas model *extends an ideal personas model* $PM = (Q, T, \Lambda, \Phi)$ *by not restricting the transition relation, and by adding a probability mapping function* $P : T \to [0, 1]$*, attaching probabilities to transitions, with the property that*

$$\forall q \in Q, \alpha \in \Lambda : \Sigma_{q \xrightarrow{\alpha} q' \in T} P(q, \alpha, q') = 1.$$

The way we defined our probabilistic model has been studied under the name of *reactive* probabilistic models in [12,19], where the Markov decision processes would fit, as opposed to *generative* models [12]. To explain the difference we first need some notation, taken from [37].

Notation 3.6. *A* discrete probability distribution *over some set Ω is a function $\mu : \Omega \to [0,1]$ such that the (support) set $\{x \in \Omega \mid \mu(x) > 0\}$ is finite and $\sum_{x \in \Omega} \mu(x) = 1$. We denote by $\mathcal{D}(\Omega)$ the set of all such distributions over Ω.*

Our probabilistic personas model $PM = (Q, T, \Lambda, \Phi, P)$ can be seen as assigning to each persona $q \in Q$ and action $\alpha \in \Lambda$ a probability distribution μ_q^α over the set of personas Q as follows: for each transition $q \xrightarrow{\alpha} q' \in T$ starting from q and labelled by α, assign its particular probability $\mu_q^\alpha(q') = P(q \xrightarrow{\alpha} q')$; for all other personas not reachable from q by α use the probability 0, i.e., μ_q^α resolves to 0. The restrictions in Definition 3.5 ensure that what we just defined as μ_q^α is a discrete probability distribution. This is essentially the definition of a *reactive probabilistic model*, i.e., which assigns to each $q \in Q$ a function from $\mathcal{D}(Q)^\Lambda$, i.e., which attaches to each action a distribution over Q. On the other hand, a generative probabilistic model assigns to each $q \in Q$ a distribution μ_q over the set $\Lambda \times Q$.

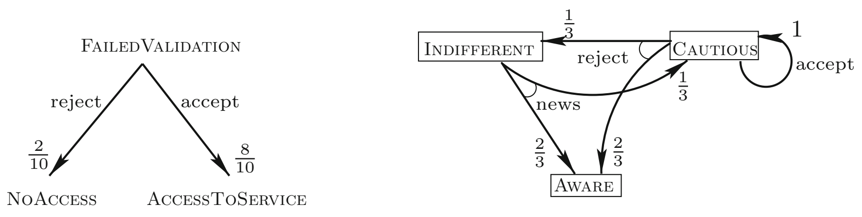
Intuitively, the generative models treat the actions as *output* of the probabilistic system, whereas the reactive models treat the actions as *input*. A reactive model takes as input an action from the environment and then probabilistically chooses the next state; whereas a generative model probabilistically chooses the next state and assigns an action to this transition, which is visible to outside.

The intuitive reason for the personas models being reactive in nature is that we want to model the knowledge that we have about how an action changes a persona. In an ideal world this should be completely determined, i.e., knowing with certain probability 1 that from a persona q an action α will result in some other persona q', i.e., for which $\mu_q^\alpha(q') = 1$. In the real world it is not so clear, but some evidence exists, and is encoded in the probabilities. Thus, from a persona, some action may result in reaching one of several personas, each with some probability. The input action can either be done intentionally by the persona, like clicking the button on the interface, or unintentional, like when being transmitted information through the adds on the metro.

On the other hand, the personas models do not capture the likelihood of some actions being taken by a persona when faced with a state of a user interface. These likelihoods will be made part of the user interface models in the next section, and will be coupled with personas through the compilations. In this way we can capture the two kinds of probabilistic information (about likelihood of changing a personas and the likelihood of choosing some action) in a single probabilistic model.

Example 3.7. In Fig. 2 we pictured a toy example of a user interface for a popup in case a certificate is not automatically validated by the browser, which offers the manual choices of *accept* and *reject*. This in turn would give access to the online service or disallow access. It is possible that the certificate is malicious, in which case acceptance may compromise the user's assets. The respective probabilities are drawn on the arrows.

The personas model studies three personas, given by the attributes in the picture (i.e., *cautious*, *indifferent*, and *aware*), and how three actions affect these



(a) UI model (generative) (b) Personas Model (reactive)

Fig. 2. Personas model and UI model.

personas (i.e., *accept*, *reject*, or *news* reading). Many other personas could be introduced in the model, like being both cautious and aware.[1]

Empirical studies may show that rejecting a certificate validation notice may make the person indifferent or more aware, each with its respective probability. Whereas reading news about some recent cyberattack done through invalid certificates may make the person more aware. Acceptance of an invalid certificate always makes someone cautious.

3.1 Models for User Interfaces

A simple model of a user interface is that of a deterministic state machine with labels on transitions denoting the options that are offered by the interface to the user in each state. The determinism is necessary because the user interface is assumed to be a piece of code running on some digital device. This code responds to the input from the user by transmitting the choice (probably in some processed form) to the back-end program and then providing a new list of choices. One can imagine this simple interface as a machine equipped with lighted buttons. The buttons that are lit are enabled, i.e., possible interaction choices for the user. Pressing a lighted button makes that the whole buttons interface changes its lighting configuration. Whereas pressing a dark button makes no change.

The above view is of a discrete nature, where interaction happens in steps, one button press at a time. But more complex interfaces can be identified, which have also forms of continuous interactions; like a car acceleration pedal, or a game console joystick. Through such interfaces the user can have a prolonged interaction with, e.g., a button, and the observable response from the interface changes with time during the user interaction. The response from the user interface is determined by the amount of time the button is pressed. Such interfaces are called *hybrid* because interaction can be both discrete and continuous. We will develop our ideas using a simple model of a user interface, and postpone the more complex hybrid models for a proper study.

[1] We draw an angle between transitions to denote those which share the same label (like *news* and *reject*); and by definition must form a probability distribution.

But just a simple deterministic model is not enough, since we are interested in some more empirical information which the personas model is not providing. We want to know for a particular user interface which actions are more probable to be taken by a persona. But coming up with this probabilistic information for each persona is infeasible. Instead we start from assumptions made by the designer of the user interface about the probability of taking each action, and rely on later empirical social studies to update these probabilities.

Definition 3.8 (User Interface Model). *We define a* user interface model *as a $UI = (S, \Lambda, T, P)$ containing a set S of states, a set Λ of actions, a transition relation $T \subseteq S \times \Lambda \times S$ which is restricted to be a partial function from $S \times \Lambda \rightarrow S$, and a probability mapping function $P : T \rightarrow [0, 1]$ with the property that*

$$\forall s \in S : \Sigma_{s \xrightarrow{\alpha} s' \in T} P(s, \alpha, s') = 1.$$

If we ignore the labels, then we have just defined a Markov chain. The action labels just say which actions change the state of the user interface. The above model should not be confused with a Markov decision process. The probabilities on transitions tell for each state which action is more probable to be executed.

Our probabilistic model for a user interface $UI = (S, \Lambda, T, P)$ can be seen as assigning to each state $s \in S$ a probability distribution μ_s over the set $\Lambda \times S$ as follows: take all transitions $s \xrightarrow{\alpha} s' \in T$ starting from s, and assign their particular probabilities $\mu_s(\alpha, s') = P(s \xrightarrow{\alpha} s')$; for all other non-existing transitions we assume to have a probability 0, i.e., μ_s resolves to 0. The restrictions in Definition 3.8 ensure that we defined μ_s as a discrete probability distribution. This is essentially the definition of a generative probabilistic model, i.e., which attaches to each $s \in S$ a distribution μ_s over the set $\Lambda \times S$. User interface models can be seen as special cases of generative models where there are no two transitions labelled by the same action (i.e., deterministic).

We need to do statistical inference, from observations, for pairs of a user interface model and personas model. This would start from a UI model which is initially populated with base rate probabilities, or even with a linear distribution, i.e., completely uninformative. The result should be a probabilistic model which should be recognized as an improved approximation of the user interface model we started with. Because of this we can apply the statistical inference over and over again, thus, more observations would imply more accurate user interface model, which will reflect better the choices of the users. We will call the inference process, the *update operation*, and the coupling of the persona and the user interface, the *compilation operation.*

4 Compilation Operations

The notion of compilation is our way of capturing the interaction between personas and user interfaces. Therefore, we do not want to be restrictive in its definition, so to allow for various future forms of interactions to be defined as compilation operations.

Definition 4.1 (Compilation). *Denote by $\mathcal{U}$ the set of all user interface models (cf. Definition 3.8) and by $\mathcal{P}$ the set of all personas models (cf. Definition 3.5). The* compilation operation $\odot$ *is defined as $\odot : \mathcal{U} \times \mathcal{P} \to \mathcal{U}$ taking a user interface model and a personas model and returning a user interface model.*

Essentially this definition requires the compilation operation to return a new user interface. This would allow for successive compilations of the result with other personas models.

We can give a more concrete definition under the following assumptions. Consider that $\mathcal{U}$ and $\mathcal{P}$ are defined over the same set of actions Λ. This requirement can be achieved for any personas model by applying (a probabilistic version of) the action restriction operation from Definition 3.4 on the set of labels of the user interface model. Consider only those personas models which have a *total* transition function, i.e., for each persona all the actions are possible, leading to some other persona; call this a *total personas model.* We can view a general personas model as a total personas model where all those actions that do not exist in the general model are self-loops, having origin and target the same persona, and probabilities 1. A minimal total personas model consists of a single persona with a self-looping transition for each action. Other kinds of total personas models can be thought, e.g., with the missing actions added in a completely non-deterministic manner, i.e., using uniform distributions, meaning that noting is known about how these actions change the personas. But we are not concerned with these in the following definition.

Definition 4.2 (Concrete Compilation). *For a user interface model $UI = (S, \Lambda, T_U, P_U)$ and a total personas model $PM = (Q, T_P, \Lambda, \Phi, P_P)$ over the same set of actions Λ, we define $\odot_c$ as:*

$$UI \odot_c PM \overset{def}{=} (S', \Lambda, T', P'), \text{ where}$$

1. $S' = S \times Q$,
2. $((s_1, q_1), \alpha, (s_2, q_2)) \in T'$ *iff* $(s_1, \alpha, s_2) \in T_U \wedge (q_1, \alpha, q_2) \in T_P$,
3. $P'((s, q), \alpha, (s', q')) = P_U(s, \alpha, s') \cdot P_P(q, \alpha, q')$.

We have essentially made a synchronous product. The use of synchronous product is natural here because we consider that one action in the user interface should be actually triggered by the user, therefore the same action should be visible on the personas model. In the personas model this interaction with the user interface may trigger a change of persona, but not necessarily.

Looking more closely at the concrete compilation operation from above we can see that the transitions from the user interface model are preserved. This happens because we worked only with total personas models.

Proposition 4.3. *The compilation $\odot_c$ returns a generative probabilistic model (i.e., like a UI model). This model may contain non-determinism (i.e., several transitions with the same action) since the personas models contain these.*

Proof. This is easy to see because for each resulting pair we multiply probabilities from the reactive model, which sum up to 1 for each action, with the generative probabilities. This results in a probability distribution over the new transitions because the whole sum equals to 1.

There are drawbacks with the above concrete compilation, mainly coming from the restrictions we imposed. We cannot model interactions in the layer III where the persona does not observe actions of the UI, and thus neither changes of state in UI. Internal actions are obvious examples, but other actions may be deliberately hidden by some attacker. Also there may be actions on the personas model which we want to not be correlated with the model. Either such actions come from outside layer III, and influence it, or are actions from the persona that fail to trigger the appropriate response on the UI. A third aspect is the complete match of actions which Definition 4.2 requires. This is not usual in practice, but rather an approximation is made by the designer of the interface about what the UI actions are associated to in the mind of the user. These are not always good approximations, and could be learned from empirical studies.

Definition 4.4 (General Compilation). *For a user interface model* $UI = (S, \Lambda_U, T_U, P_U)$ *and a total personas model* $PM = (Q, T_P, \Lambda_P, \Phi, P_P)$*, and a total function* $g : (\Lambda_U \times \Lambda_P) \rightarrow \Lambda'_*$*, we define* $\odot_g$ *as:*

$$UI \odot_g PM \overset{def}{=} (S', \Lambda', T', P'), \text{ where}$$

1. $S' = S \times Q$,
2. $((s_1, q_1), \alpha', (s_2, q_2)) \in T'$ *iff* $(s_1, \alpha_u, s_2) \in T_U \wedge (q_1, \alpha_p, q_2) \in T_P$ *and* $g(\alpha_u, \alpha_p) = \alpha' \neq *$,
3. $P'((s, q), \alpha, (s', q')) = P_U(s, \alpha_u, s') \cdot P_P(q, \alpha_p, q')$.

The compilation $\odot_g$ is parametrized by the function g. When choosing the $g(\alpha_u, \alpha_p) = \alpha$ iff $\alpha_u = \alpha_p = \alpha$, and equals $*$ for all other cases, we find the particular definition of concrete compilation. Care needs to be taken when defining the g parameter because depending on this is whether we obtain a probabilistic system, and what kind, after the compilation. In such cases variations of the general compilation would be devised which would take into account the specific definition of g when building the probabilities.

The general compilation still has one drawback which does not have an easy solution. The fact that the personas model is compiled with a single UI model means that for each persona we would have the same actions likelihoods provided by this single UI. But in reality one persona (e.g., a cautious one) coupled with one UI would incur different likelihoods for the provided actions than would be the case for another persona (e.g., an indifferent one).

This drawback is important because it has been argued that each protocol, in our case each user interface, behaves differently when put in different contexts, in our case when coupled with different personas (i.e., not only different users, but the same user in different social and cognitive contexts). This is the purpose of security ceremonies, to analyse a protocol in all its contexts. This view is similar to how the symbolic approach to verification of security protocols is working; there, observational equivalence is used as a tool, thus allowing them to analyse a protocol w.r.t. all possible contexts definable in some protocol language. The compilation should allow for such kind of reasoning too.

For the update operation we can use a simple statistical inference method applicable to Markov chains [5] (or probabilistic automata [16,30]), like maximum likelihood estimation [39]. For a persona (or personas model) and a user interface model consider a *test* to be a finite sequence of actions part of a run of the compilation of the persona and the user interface. Several such tests form a *test set*. The update operation takes a compilation of a personas model and a user interface model, together with a test set, and applies a statistical inference method to obtain a new user interface model with the probability function updated according to the tests.

5 Conclusion and Further Work

For security ceremonies the abstract model introduced in [3] (which we called the Bella-Coles-Kemp model) provides a nice setup for developing formal models and analysing security breaches by concentrating on the different parts of a ceremony. We concentrated here on the human-computer interaction which forms layer III. In particular, we have argued for a probabilistic model of the human personas interacting with a user interface. The probabilities are supposed to be inferred from statistical tests, and should give a more realistic view on how a human provides input to (and receives information from) a security protocol through the user interface.

We have separated the model of the humans from the model of the user interface, each capturing some different information natural for the respective model. In particular, the personas model is reactive in nature, capturing the probabilistic knowledge about how an action affects the persona. Whereas the user interface model is generative in nature, capturing the probabilistic information about which choices are more probable to be taken at each state of the UI. These information are put together through compilation, resulting in a probabilistic model that can be used at both layers II and IV.

Probabilistic models are more close to reality, incorporating more knowledge usually obtained through empirical studies. A wealth of analysis techniques exist based on probabilistic models. These usually give quantitative answers (besides also standard qualitative ones). One example is model checking [24] based on probabilistic versions of temporal logics. A mature tool for this is PRISM, actively developed at Oxford [18].

Examples of probabilistic reasoning about security ceremonies using the framework we just proposed, are the subject of a different paper. But very common properties would be defined as reachability problems. Reachability gives quantitative answers to whether a particular situation is reached and with what probability. In our invalid certificate example one would be interested in what are the probabilities for whether the user would accept an invalid certificate or would reject a valid certificate (i.e., not recognized by the browser).

We have only looked at models and how they appear to capture probabilistic aspects of the human in security ceremonies. We have not looked at languages for describing such models. Such a language would be needed before being able to apply a tool like PRISM. The same as labelled finite state systems (used to model programming systems) have a wealth of process algebras as languages of varying abstraction capabilities, probabilistic models also have probabilistic process algebras [17]. Since the models we described are close to existing probabilistic automata, the best approach would be to start looking for existing probabilistic process algebras that we could use. What we might need is to add the compilation operations as operators of the process algebra.

References

1. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Hankin, C., Schmidt, D. (eds.) POPL, pp. 104–115. ACM (2001)
2. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: the spi calculus. Inf. Comput. **148**(1), 1–70 (1999)
3. Bella, G., Coles-Kemp, L.: Layered analysis of security ceremonies. In: Gritzalis, D., Furnell, S., Theoharidou, M. (eds.) SEC 2012. IFIP AICT, vol. 376, pp. 273–286. Springer, Heidelberg (2012)
4. Bevan, N.: International standards for HCI and usability. Int. J. Hum. Comput. Stud. **55**(4), 533–552 (2001)
5. Billingsley, P.: Statistical Inference for Markov Processes. The University of Chicago Press, Chicago (1961)
6. Blanchet, B.: Automatic proof of strong secrecy for security protocols. In: IEEE Symposium on Security and Privacy, pp. 86–102. IEEE Computer Society (2004)
7. Blanchet, B.: A computationally sound mechanized prover for security protocols. IEEE Trans. Dependable Sec. Comput. **5**(4), 193–207 (2008)
8. Carlos, M.C., Martina, J.E., Price, G., Custódio, R.F.: An updated threat model for security ceremonies. In: Shin, S.Y., Maldonado, J.C. (eds.) 28th Annual ACM Symposium on Applied Computing (SAC 2013), pp. 1836–1843. ACM (2013)
9. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–207 (1983)
10. Ellison, C.: Ceremony design and analysis. Cryptology ePrint Archive, Report 2007/399 (2007)
11. Ferreira, A., Giustolisi, R., Huynen, J.L., Koenig, V., Lenzini, G.: Studies in socio-technical security analysis: authentication of identities with TLS certificates. In: TrustCom/ISPA/IUCC, pp. 1553–1558. IEEE (2013)
12. van Glabbeek, R.J., Smolka, S.A., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Inf. Comput. **121**(1), 59–80 (1995)
13. Goldsmith, M., Lowe, G., Roscoe, B., Ryan, P., Schneider, S.: Modelling and Analysis of Security Protocols. Pearson Education, Harlow (2000)
14. Groote, J.F., Mathijssen, A., Reniers, M.A., Usenko, Y.S., van Weerdenburg, M.: The formal specification language mCRL2. In: Methods for Modelling Software Systems (MMOSS 2006). Dagstuhl Seminar Proceedings, vol. 06351 (2007)
15. Harel, D., Tiuryn, J., Kozen, D.: Dynamic Logic. MIT Press, Cambridge (2000)
16. de la Higuera, C., Oncina, J.: Learning stochastic finite automata. In: Paliouras, G., Sakakibara, Y. (eds.) ICGI 2004. LNCS (LNAI), vol. 3264, pp. 175–186. Springer, Heidelberg (2004)

17. Jonsson, B., Larsen, K.G., Yi, W.: Probabilistic extensions of process algebras. In: Bergstra, J., Ponse, A., Smolka, S. (eds.) Handbook of Process Algebras, pp. 685–711. Elsevier, Amsterdam (2001)
18. Kwiatkowska, M., Norman, G., Parker, D.: Advances and challenges of probabilistic model checking. In: Proceedings of the 48th Annual Allerton Conference on Communication, Control and Computing, pp. 1691–1698. IEEE Press (2010)
19. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Inf. Comput. **94**(1), 1–28 (1991)
20. Latour, B.: Reassembling the Social - An Introduction to Actor-Network-Theory. Oxford University Press, Oxford (2005)
21. Lowe, G.: Breaking and fixing the needham-schroeder public-key protocol using FDR. Softw. Concepts Tools **17**(3), 93–102 (1996)
22. Mitchell, J.C., Mitchell, M., Stern, U.: Automated analysis of cryptographic protocols using Murphi. In: IEEE Symposium on Security and Privacy, pp. 141–151. IEEE Computer Society (1997)
23. Newell, A.: Unified Theories of Cognition. Harvard University Press, Cambridge (1990)
24. Norman, G., Parker, D., Sproston, J.: Model checking for probabilistic timed automata. Formal Meth. Syst. Des. **43**(2), 164–190 (2013)
25. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. J. Comput. Secur. **6**(1–2), 85–128 (1998)
26. Pavlovic, D., Meadows, C.: Actor-network procedures: modeling multi-factor authentication, device pairing, social interactions. arXiv.org (2011)
27. Pieters, W.: Representing humans in system security models: an actor-network approach. J. Wirel. Mob. Netw. Ubiquit. Comput. Dependable Appl. **2**(1), 75–92 (2011)
28. Pratt, V.R.: Process logic. In: 6th Symposium on Principles of Programming Languages (POPL 1979), pp. 93–100. ACM (1979)
29. Prisacariu, C.: Actor network procedures as psi-calculi for security ceremonies. In: International Workshop on Graphical Models for Security. Electronic Proceedings in Theoretical Computer Science, vol. 148, pp. 63–77. Open Publishing Assoc. (2014)
30. Rabin, M.O.: Probabilistic automata. Inform. Control **6**(3), 230–245 (1963)
31. Radke, K., Boyd, C., Gonzalez Nieto, J., Brereton, M.: Ceremony analysis: strengths and weaknesses. In: Camenisch, J., Fischer-Hübner, S., Murayama, Y., Portmann, A., Rieder, C. (eds.) SEC 2011. IFIP AICT, vol. 354, pp. 104–115. Springer, Heidelberg (2011)
32. Rogers, Y., Sharp, H., Preece, J.: Interaction Design: Beyond Human-Computer Interaction, 3rd edn. Wiley, Chichester (2011)
33. Rukšėnas, R., Curzon, P., Back, J., Blandford, A.: Formal modelling of cognitive interpretation. In: Doherty, G., Blandford, A. (eds.) DSVIS 2006. LNCS, vol. 4323, pp. 123–136. Springer, Heidelberg (2007)
34. Ruksenas, R., Curzon, P., Blandford, A.: Modelling and analysing cognitive causes of security breaches. Innovations Sys. Softw. Eng. **4**(2), 143–160 (2008)
35. Segerberg, K.: Getting started: beginnings in the logic of action. Stud. Logica **51**(3/4), 347–378 (1992)
36. Semančík, R.: Basic properties of the persona model. Comput. Inform. **26**(2), 105–121 (2007)

37. Sokolova, A., de Vink, E.P.: Probabilistic automata: system types, parallel composition and comparison. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 1–43. Springer, Heidelberg (2004)
38. Stern, U., Dill, D.L.: Parallelizing the Murφ verifier. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 256–278. Springer, Heidelberg (1997)
39. Teodorescu, I.: Maximum likelihood estimation for markov chains (2009). arxiv:0905.4131