Models-Based Analysis of Both User and Attacker Tasks: Application to EEVEHAC

Sara Nikula, Célia Martinie, Philippe Palanque, Julius Hekkala, Outi-Marja Latvala, Kimmo Halunen

ModelS-based Analysis of both User and Attacker Tasks: Application to EEVEHAC

Sara Nikula[1][0000−0002−2299−8030], Célia Martinie[2][0000−0001−7907−3170], Philippe Palanque[2][0000−0002−5381−971X], Julius Hekkala[1][0000−0002−7558−9687], Outi-Marja Latvala[1][0000−0001−8083−8986], and Kimmo Halunen[3][0000−0003−1169−5920]

[1] VTT Technical Research Centre of Finland, Kaitoväylä 1, 90571 Oulu, Finland
firstname.lastname@vtt.fi
[2] ICS-IRIT, Université Toulouse III - Paul Sabatier, Toulouse, France
lastname@irit.fr
[3] University of Oulu, Finland
Kimmo.Halunen@oulu.fi

Abstract. The design and development of security mechanisms, such as authentication, requires analysis techniques that take into account usability along with security. Although techniques that are grounded in the security domain target the identification and mitigation of possible threats, user centered design approaches have been proposed in order to also take into account the user's perspective and needs. Approaches dealing with both usability and security focus on the extent to which the user can perform the authentication tasks, as well as on the possible types of attacks that may occur and the potential threats on user tasks. However, to some extent, attacker can be considered as user of the system (even if undesirable), and the analysis of attacker tasks provides useful information for the design and development of an authentication mechanism. We propose a models-based approach to analyse both user and attacker tasks. The modeling of attacker tasks enables to go deeper when analysing the threats on an authentication mechanism and the trade-offs between usability and security. We present the results of the application of this models-based approach to the EEVEHAC security mechanism, which enables the setup of a secure communication channel for users of shared public computers.

Keywords: Task modeling · Usable security · Human understandable cryptography · Visual channel

1 Introduction

Security mechanisms have an impact on human performance because they add additional activities to users that do not correspond to their main goals [28]. For example, when the main purpose of a user is to check a bank account statement on a website, the user will not directly consult the statement after having entered the service web page. Before that, the user will have to authenticate and

thus to perform additional actions that aim to grant access to the bank account statement. These additional activities require to engage additional resources (e.g. temporal, cognitive, motor...) and decrease the user global performance. However, these mechanisms may be necessary when they correspond to threats to be avoided. In our example, the authentication mechanisms aims to avoid that user's data be stolen, compromised, destroyed or used for malicious purposes. All of the authentication mechanisms do not have the same impact on user tasks, and thus are not equal in terms of level of usability. In the same way, all of the authentication mechanisms are not equal in terms of level of security. When designing authentication mechanisms, both usability and security aspects have to be taken into account [7], in order to explore the possible trade-offs, as well as to perform informed design choices. Existing research work on the engineering of usable and secure authentication mechanisms focuses on the extent to which the user can perform the authentication tasks (using empirical or analytical approaches), as well as on the possible types of attacks that may occur and the potential threats on user tasks [7]. It is acknowledged that a specific feature of an authentication mechanism may have an important impact on user's tasks, and as a consequence, trigger its integration or removal. But a specific feature of an authentication mechanism may also have an important impact on attacker's tasks, either making them almost impossible or trivial to perform. To some extent, attacker can be considered as (undesirable) user of the system, and the authentication mechanism should be designed to make impossible the attacker's task.

This paper presents a modelS-based approach to analyse both user and attacker's tasks when designing and developing an authentication mechanism. The capital S in the end of the word model stands for the different types of models required to apply the proposed approach. It combines task models, to describe explicitly user and attacker's tasks, and attack tree models to describe explicitly alternative paths of attacks. The article is organized as follows. Section 2 introduces the main theoretical aspects of the proposed approach. Section 3 presents the two types of models required for the application of the proposed approach and how they complement each other. Section 4 presents the results of the application of the proposed approach to the EEVEHAC authentication mechanism. Section 5 presents the related work on methods and techniques to usable and secure authentication mechanisms. Section 6 concludes the paper.

2 Towards Humans Centric Security Engineering

The engineering of usability and security require to take into account the user tasks to ensure usability and to identify potential security threats on these tasks. But users may not be the only humans involved while the authentication system executes. An attacker may also interact with the authentication system and thus authentication mechanisms design and development approaches have to deal with attacker tasks.

2.1 Generic requirements for engineering authentication mechanisms for usability and security

The ISO standards defines usability as *"the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"* [19]. The analysis of effectiveness requires to identify precisely and exhaustively user tasks, in order to check that all of them are supported by the authentication mechanism [23]. Moreover, ensuring effectiveness requires to check that the appropriate function is available at the time when the user needs it. The analysis of efficiency also requires a precise identification and description of user tasks. In the case of predictive assessment of efficiency (e.g. Keystroke Level Model prediction techniques [10] [13] [18]), the task types and their temporal ordering are required to be identified in order to associate a standard predicted value and to calculate the estimated time. In the case of empirical assessment of efficiency, tasks also have to be identified to prepare the user testing protocol. The analysis of satisfaction requires user feedback because it is a subjective criteria that relies on the individual characteristics of the user, though it is dependent on effectiveness and efficiency. The analysis of satisfaction also requires the identification of user tasks as the user feedback may be referring to a particular task [4].

Security analysis highly relies on the identification and description of potential threats [9]. The analysis of possible threats on user tasks requires precise and exhaustive description of user tasks [7]. In particular, it requires to identify the user task types (a threat can arise from a type of user action, e.g. drawing a gesture password on a tactile screen is subject to smudge attacks whereas typing a password on a keyboard is subject to key-logging attack), their temporal ordering (a threat can arise from the specific ordering of user tasks, e.g. leave the credit card in the automated teller machine once having withdrawn the notes), and the information, knowledge, objects and devices being manipulated while performing the tasks (a threat can arise from an information, knowledge or object that the user has lost, forgotten or misused, e.g. a credit card lost in a public space).

2.2 Generic requirements for engineering the attacker tasks to be as complex as possible

The analysis of complexity of attacker tasks requires the precise and exhaustive identification of the tasks to attack the authentication mechanism. Precise and exhaustive description of tasks enable to analyse effectiveness and efficiency, and as such to ensure a very low level of effectiveness and efficiency for an attacker. In the same way that the analysis of effectiveness and efficiency requires the identification of the user tasks types, of their temporal ordering, as well as of the information, knowledge, objects and devices manipulated while performing the tasks, the analysis of the attacker tasks also does.

3 Models-based analysis of both user's and attacker's tasks

The production of task models of both user and attacker tasks enable to systematically and exhaustively analyse the effectiveness and efficiency of the user and of the attacker with an authentication mechanism, provided that the task modeling notation is expressive enough to fulfill the requirements presented in the previous section.

3.1 Task model based analysis of user tasks using HAMSTERS-XL

Task models consist of a graphical, hierarchical and temporally ordered representation of the tasks the users have to perform with an interactive application or system in order to reach their goals. Task models are one of the very few means for describing user tasks explicitly and exhaustively [22]. Task models support several different stages of interactive systems design and development (e.g. user roles identification, system functions identification, user interface design, testing, training program design...). We selected the HAMSTERS-XL notation [22] because it enables to describe the required types of tasks such as user task (cognitive, perceptive, and motor), abstract tasks, interactive tasks (input, output), and system tasks, as well as their temporal ordering and the information, knowledge, objects and devices manipulated while performing the tasks.

3.2 Attack tree based analysis of possible attacks

An attack tree [29] describes the possible attacks or combination of attacks on a system. It is composed of a main goal for an attack, which is represented by the top root node, and of a combination of leaves that represent different ways to achieve that goal. In the original notation [29], OR nodes (presented in Figure 1) refer to alternatives of attacks to achieve the attack whilst AND nodes refer to combination of attacks. The notation has been extended to enable the description of the potential effects of attacks, as well as to enable the description of the combination of attacks, using the SAND logical operator. Other elements of the notation include a rectangle to represent an event (such a threat or attack), an ellipse to represent an effect and a triangle to represent the fact that a node is not refined. All elements of the attack tree notation are shown in Figure 1. Attack trees enable to systematically identify possible attacks on an authentication mechanism.

3.3 Task model based analysis of attackers' tasks

An attack tree describes attack goals, but does not describe the possible tasks and their temporal ordering to reach these attack goals. It thus presents a partial view on attacker's tasks. Although attack trees represent the main attack goal and its associated possible combination of ways to reach the main goal of the

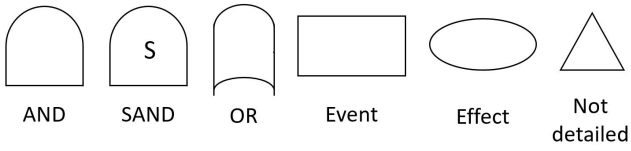


Fig. 1. Elements of notation for the attack trees from [27]

attack, the attacker tasks to reach an attack goal may be very different from the tasks to reach another attack goal. For example, the tasks to perform a video attack are very different from the tasks to perform a shoulder surfing attack. In the first case, the attacker has to identify a location where the camera could be installed, as well as means to either trigger the recording at the appropriate time or to extract the video sample when the user was authenticating if the record covers a long time period. Several specific preparation tasks are required and specific devices are required too. In the other case, the attacker has to stand behind the user at the right time, but does not need special devices.

Task models of attackers tasks that provide exhaustive description of the tasks as well as of information, data, objects and devices thus help to identify whether the authentication mechanism is worth to implement, by explicitly highlighting the complexity of the attack.

4 Validation of the approach: Models-based analysis of usability and security of EEVEHAC

In this section, we present the results of the application of the proposed approach for the analysis of the EEVEHAC (End-to-End Visualizable Encrypted and Human Authenticated Channel) secure communication channel mechanism [17].

4.1 EEVEHAC: End-to-End Visualizable Encrypted and Human Authenticated Channel

EEVEHAC uses human understandable cryptography and has been designed to enable users to safely recover and use sensitive private data (e.g. bank account statement) when using public untrusted devices (e.g. public desktop in libraries or cybercafés). EEVEHAC establishes a secure communication channel and warns the user about the possibility of corruption of the public device if necessary. EEVEHAC is actually implemented for smartphones but targets smaller wearable devices such as smart glasses.

Overview of EEVEHAC. Modern cryptographic mechanisms are based on complicated mathematics that regular human users cannot understand. However, for a regular user, the common experience of using it is completely opaque: after

they type in their password, there is no further interaction and no indication of whether the cryptographic protocols behind the scene have executed in a correct manner and without interference from attackers. Because cryptography is mathematically complicated, it is fully done by machines and there is no immediate way for a user to know if the results are correct or corrupted. In order to mitigate this issue, there has been some previous research in developing human understandable cryptography. An early example of this is visual cryptography [26], where the user can decrypt a secret message simply by looking at it. A recent review [16] presents more examples of the use of human abilities in cryptographic systems. There are certain key building blocks that utilize human capabilities in cryptography, but no complete end-to-end communications systems existed, until EEVEHAC was proposed [17]. EEVEHAC composes of two security protocols: HAKE (Human Authenticated Key Exchange) [5] and EyeDecrypt [15]. The user first sets up a long term key, also referred to as "the long time secret", with a trusted server. Based on this long term secret, the user then setups of a secured communication channel using a smartphone which implements the HAKE protocol. The second security protocol, EyeDecrypt, provides a visual channel to communicate securely using a public terminal. By visual channel, we mean that it provides to the user visual indication about the possible corruption of the communication channel.

Main steps for configuring the long term key. The steps for the configuration of the long term key are presented in Table 1. The authentication information contains a story and a mapping between six colors and numbers, which the user needs to memorize. These compose the long term key and are acquired during registration to the service. The story is mostly computer generated, but the user has an option to change one word for each sentence. The intention is to balance the strengths and weaknesses of both machine and human generated stories [30]. Once the long term key (story and color code) is configured, the server and the user's trusted device (smartphone) have matching keys (AES [11] and HMAC [20]) and the second protocol of EEVEHAC can be used.

Main steps to log in to a service including the set up of a secure communication channel. The steps for initiating a secure communication channel in order to log in to a service, i.e. performing a HAKE protocol leading to UAN (Unique Authentication Number) code, are presented in Table 2. The smartphone authentication application presents sentences from the original story where one word is replaced, so that the sentence remains grammatically correct but the meaning is changed. The user needs to spot the changed words and take note of their colors. They then recall the corresponding numbers, count their sum and use the modulo 10 of the sum as the first digit of the UAN. This process is iterated 3 times more (one iteration for each digit of the UAN code which is 4 digit long). Once the UAN code is entered and validated by the smartphone application, the secure channel is ready and the user can log in to the target service (e.g. bank account). Table 3 presents the steps to log in the target service

Table 1. Main steps for configuring the long term key.

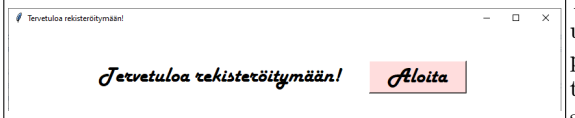
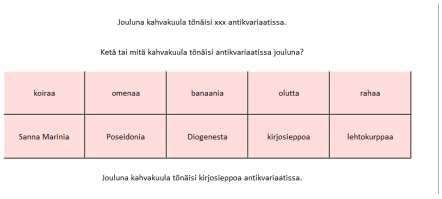
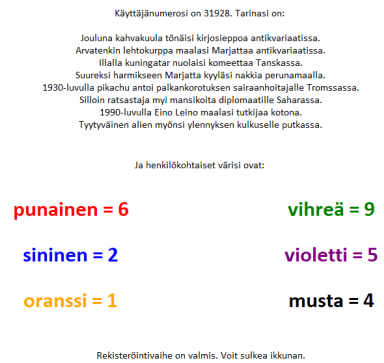
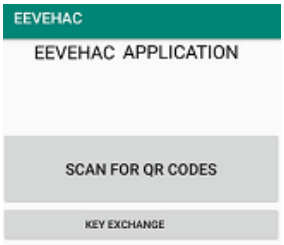
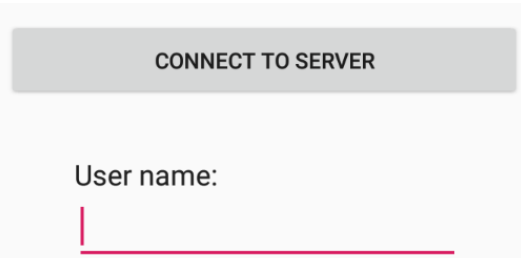
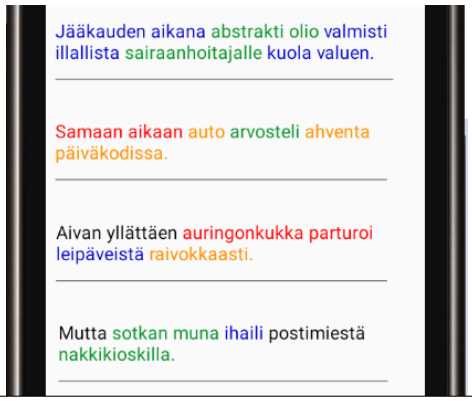
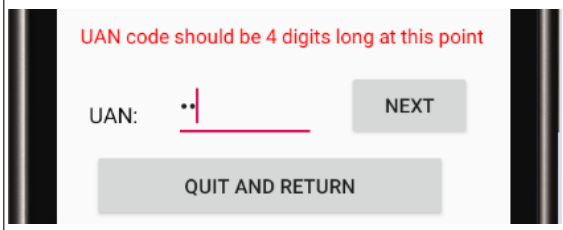
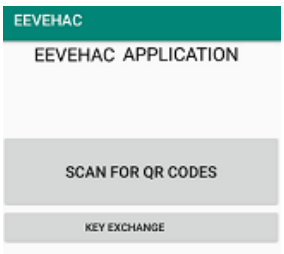
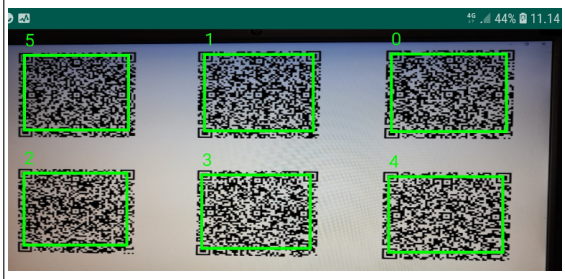
Step	Screenshot of the application	Description
1		This screen welcomes the user to the registration process. The user clicks the light red button to start the process.
2		Instructions appear on the screen. The user is supposed to fill in a missing word in each sentence.
3		First sentence appears on the screen. The uppermost sentence is to be filled. Alternative words are shown on the light red buttons. The user picks one of these by clicking it. The filled sentence is shown on the bottom line. This is repeated eight times.
4		After the user has completed all sentences, user specific user number is shown on the uppermost row. The user specific story, consisting of the eight sentences completed in the previous step, is shown below. The user specific color-number-mapping is shown on the colored lines: the number corresponding to red is 6, the number corresponding to blue is 2, and so on. The last row states that the registration phase is ready and the user can close the window.

Table 2. Main steps to setup the secure communication channel.

Step	Screenshot of the application	Description
1		The user opens the application and clicks the key exchange button to start the HAKE protocol.
2		The user fills in their user number.
3		The application connects to the server and starts the HAKE protocol. The user must first detect which two sentences belong to the story and which words have been changed. The user then recalls the numbers corresponding to colors of these words. They then sum these colors and count modulo 10 of this sum.
4		The user inputs the result in the UAN field shown under the sentences. After every set of sentences, the user inputs one number, the length of the UAN code increases by one after every screen. This is done four times. After the last screen, the user is either informed that the protocol was successful or that they must try again.

using the secured channel. The server sends encrypted messages to the untrusted device (public desktop), which shows them in their respective grid positions as QR codes. The user points the camera of their smartphone to the screen of the laptop to scan the grid. The EEVEHAC application processes the QR codes, checking their position and trying to decrypt the contents. Correct positioning of the codes is indicated with green outlines on top of the camera feed. Red outlines are used for incorrect positions. Uncorrupted QR can be successfully decrypted and the results shown to the user in the camera feed. Conversely, if the application cannot process the QR codes and show the results, the user can deduct that something has gone wrong in the process and that an attack may be going on. By referencing the camera feed, the user can now input a short PIN code through the untrusted device to access the underlying service provided by the server.

Table 3. Main steps to input service PIN code while using the visual channel by scanning QR codes.

Step	Screenshot of the application	Description
1		The user clicks the "Scan QR codes" button.
2		The user sees QR codes on the terminal and points at them with the camera of their mobile device. If green rectangles appear around the QR codes, the user knows that everything is OK. Based on the decrypted content shown in the application the user can select the right QR code.

4.2 Task models of user tasks with EEVEHAC

In this section, we present the main task models that were produced to analyse usability of EEVEHAC. For the explanation of the application of the approach, we present the high-level parts of the task models. We also focus on specific

branches of the task models, for which we present the details of the tasks, in order to highlight how we use them for the analysis presented in section 4.5.

Figure 2 presents the task model that describes the main user tasks to configure the long term key. These consist of building (with application support) a story, memorizing the story and memorizing the color codes. The task model reads from left to right and from top to bottom. The main goal is represented by the abstract task at the top of the model "Configure long term key (story and color code)". To reach this main goal the user has to perform a sequence of four sub-goals, also represented by abstract tasks. All of the abstract tasks are refined in concrete user tasks. For example, the user first starts the configuration (abstract task "Start configuration" on the left of the model). This abstract task decomposes in a sequence ('>>' operator) of the cognitive task "Decide to start", the interactive input task "Click on start" and the interactive output task "Display welcome message". The abstract task "Select a word for the story" is iterative. The user manipulates several information (green boxes) to configure the long term key (e.g. story, value for orange color...). In the end, the user has to memorize this information, which is described by the production of declarative knowledge (represented with violet boxes names "Story" and "color codes").

Figure 3 presents the task model that describes the main user tasks to log in to the service with EEVEHAC. Under the main goal labelled "Log in to the service" at the top of the task model, a sequence of tasks has to be accomplished (temporal ordering operator "sequence" represented with ">>"). The main high-level sub goals are, that first, the user starts the service provider application on his smartphone, then, the user performs the UAN authentication and the last subgoal is to insert the PIN code. The abstract task "Select QR code" is represented as folded (with the symbol '+' at the bottom right of the task name). This means that this tasks decomposes in several tasks (not presented here).

4.3 Attack tree

In this section we focus on an extract of the attack tree of EEVEHAC, for which we present the details of the attack, in order to highlight how we used it for the analysis presented in section 4.5. A full description and a threat analysis of the EEVEHAC system is presented in [17].

Figure 4 presents the extract of the attack tree of EEVEHAC for the attacker goal "Impersonate the server or the user". The attacker goal may be to impersonate the server or the user (rectangle labelled "Impersonating the server or the user" on the top of the attack tree). This attack may have 3 effects (3 ellipses on the top of the attack tree): the communication channel is compromised, sensitive user data is stolen and the user device is infected with malware. If the attacker gets access to the encryption key, which is used to encrypt and decrypt the communication channel, the channel is no more secure. To reach the attack goal, the attacker must conduct a man-in-the-middle attack and uncover the long term key (story and color code). This is represented with the two rectangles under the "AND" operator. The man-in-the-middle attack is very unlikely because the attacker must simultaneously impersonate trusted devices (the server and the user

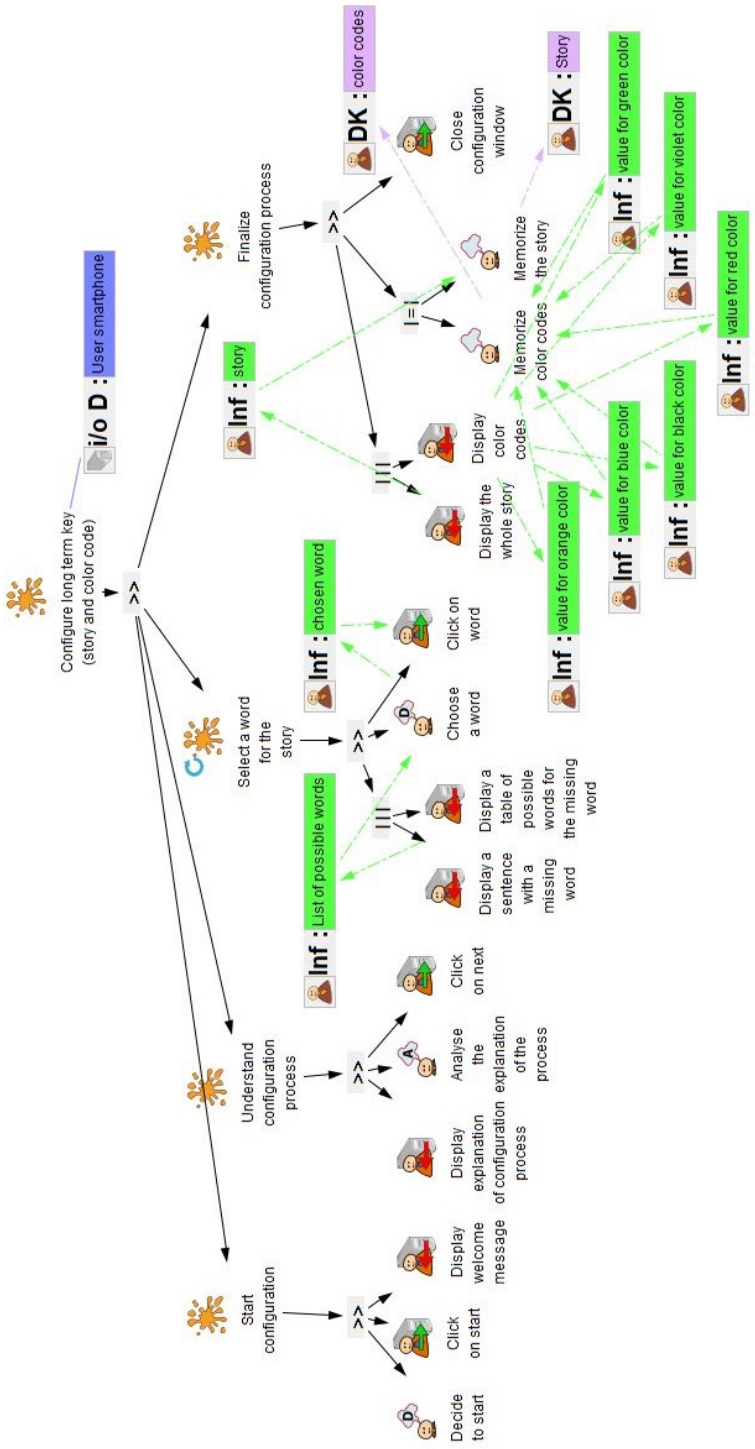


Fig. 2. Task model describing the task to "Configure the long term key (story and color code)"

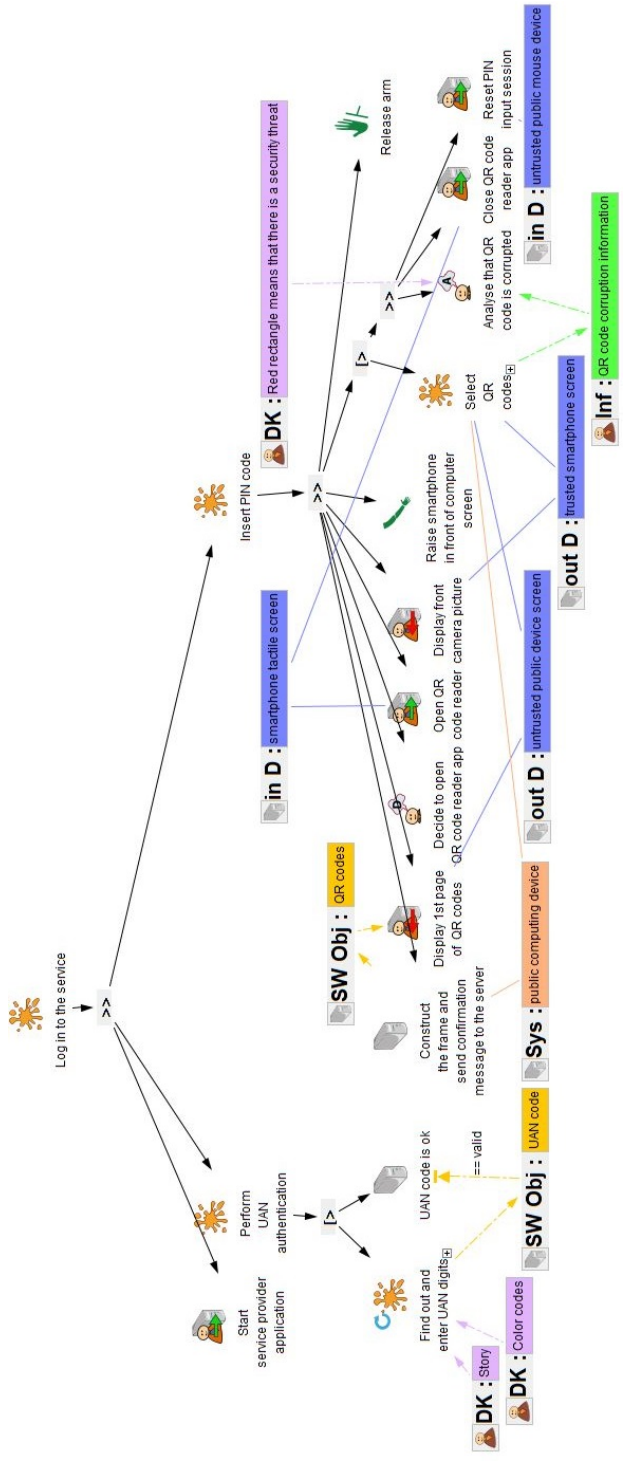


Fig. 3. Task model describing the task to "Log in to the service"

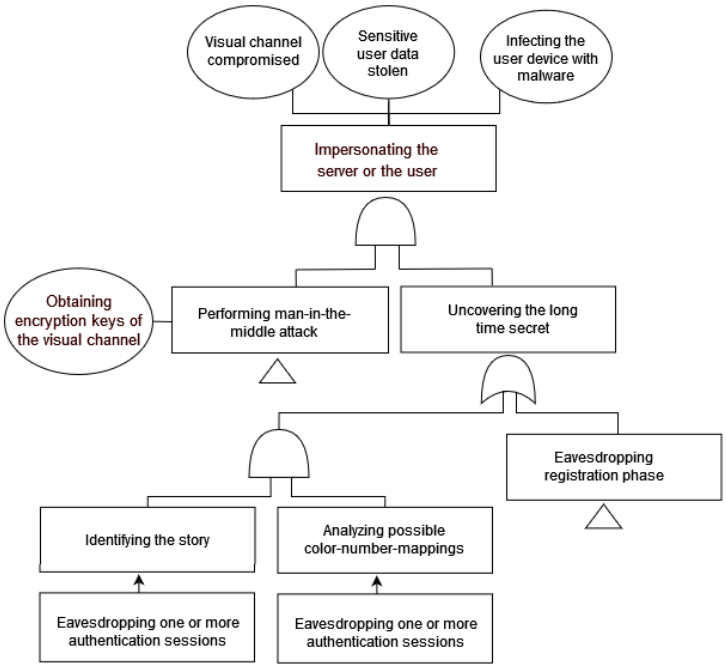


Fig. 4. Attack tree for the attacker goal "Impersonating the server or the user"

smartphone). This attack is thus not detailed (represented with a triangle). To uncover the long term secret, the attacker can either steal the information during the registration phase (rectangle labelled "Eavesdropping registration phase") or ("OR" operator) eavesdrop HAKE protocol communication channel setup sessions to identify the story and the color code (rectangles labelled "Identifying the story" and "Analyzing possible color-number-mappings" under the "AND" operator). Because the registration (configuration of the long term key) is supposed to be done in a peaceful place, such as home, it is highly unlikely that the information is eavesdropped in this phase. This attack is thus not detailed (represented with a triangle).

4.4 Task models of attacker tasks

In this section, we present the main task models that were produced to analyse complexity of attacker tasks for the attack "Impersonating the server or the user". For the explanation of the application of the approach, we present the high-level part of the task model. We also focus on a specific branch of the task model, for which we present the details of the tasks, in order to highlight how we use them for the analysis presented in section 4.5.

Figure 5 presents the task model of the attacker to reach the goal "Uncover the long time secret by eavesdropping". This goal in the task model corresponds to the main attack goal presented in the attack tree in the previous section.

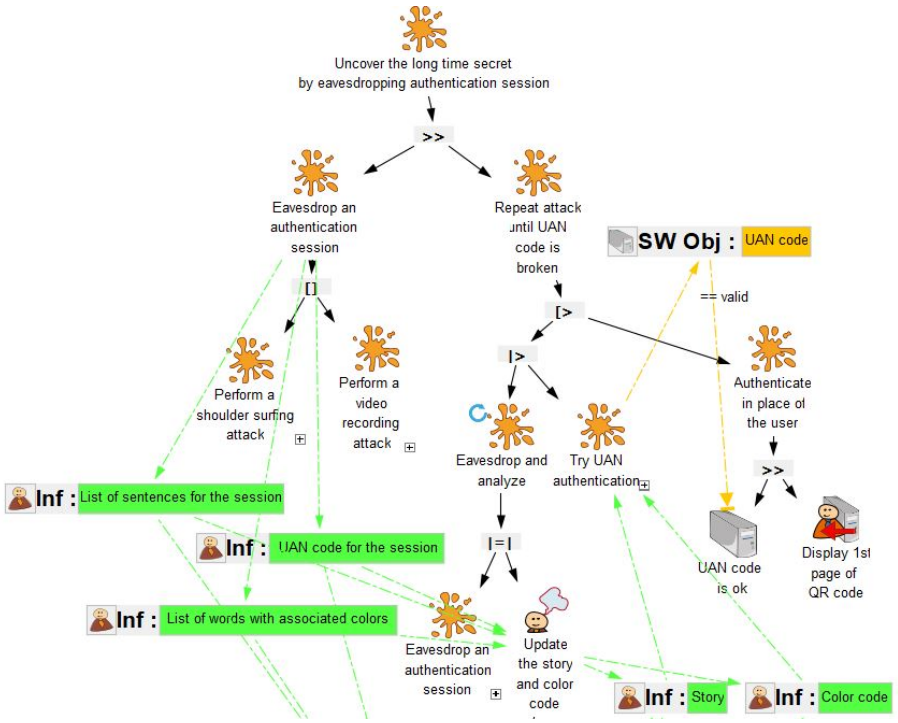


Fig. 5. Upper part (sub-goals) of the task model describing the attacker task "Uncover the long time secret by eavesdropping"

The task model presented in Figure 5 is the upper part of the whole task model for the attacker tasks. The attacker has to eavesdrop an authentication session (abstract task labelled "Eavesdrop and authentication session"), using a shoulder surfing attack or a video recording attack (folded abstract tasks labelled "Perform a shoulder surfing attack" and "Perform a video recording attack"). This first attack enables to get the first version of the list of possible sentences, as well as the corresponding UAN code and a first version of the list of possible words with their associated colors. Every iteration of this attack gives the attacker more information and thus, if the first attack is complete, the attacker has to repeat the attack until the long term key is broken (abstract task "Repeat attack until UAN code is broken"). This task decomposes in the abstract iterative task "Eavesdrop and analyze", that may be interrupted (temporal ordering operator '$|>$') by the task "Try UAN authentication". The abstract task "Authenticate in place of the user" may disable the abstract task "Try UAN

authentication" if the UAN code is valid. In this case, the attacker reached the main goal of the attack. The abstract iterative task "Eavesdrop and analyze" decomposes in the folded (not presented in detail in the paper) abstract "Eavesdrop and authentication session" and in the cognitive task "Update the story and color code". They can be performed in an order independent way (temporal ordering operator '$|=|$').

Figure 6 presents an extract of the description of the attacker task "Update the story and color code". The attacker has to perform several cognitive tasks to find out the story and the color code. The attacker has to "Identify the possible combinations of sum of colors to match each UAN digit", to "Identify wrong sentences", to "Identify correct sentences", and to "Identify color mapping". The folded tasks are not presented in detail in the paper. In order to identify the possible combinations of sum of colors to match each UAN digit, the attacker has to identify the combinations for several authentication sessions. And for each authentication session, the attacker has to identify the possible combinations for each UAN digit. And for each UAN digit, the attacker has to identify the possible combinations for the two possible number pairs that can result in the UAN digit using modulo 10. Figure 6 details the attacker tasks for 1 branch for each task. We present here a subset of the tasks because the full model does not fit in a page, and embeds more than sixty tasks, providing an evidence that the attacker will have a lot of work to perform to uncover the story and the color codes.

4.5 Task models based analysis of tradeoffs between usability and security for EEVEHAC

About the usability of EEVEHAC, we analysed the two task models: "Configure the long term key" (Figure 2) and "Log in to the service" (Figure 3). For the first one, user tasks consist of building the story and memorizing it, as well as memorizing the color codes. There are several cognitive tasks, including memory tasks. The cognitive load for choosing words for the story may not be very high, but for the memory task, from the task model, we see that several information has to be memorized (chosen words, value for orange color, value for blue color, value for black color, value for red color, value for violet color and value for green color). All of this information has to be recorded in long-term memory, and turned into declarative knowledge (color codes and story in the task model). This implies that the user has to be willing to spend time on the configuration of the long term key. However, remembering a story can be easier than remembering an arbitrary password. The work of [30] takes advantage of the human ability to better remember stories than characters and numbers and the authors suggest a system where the secret key is a story in form of a text adventure. At last, configuration of the long term key is done only once.

For the second task model, "Log in to the service" (Figure 3), the user has also to perform several cognitive tasks in order to recall and recognize the story words, identify the wrong words, recognize the colors and recall the color codes and to calculate the UAN temporary code. These tasks include cognitive calculation tasks that may take time depending on the user. This is supposed to

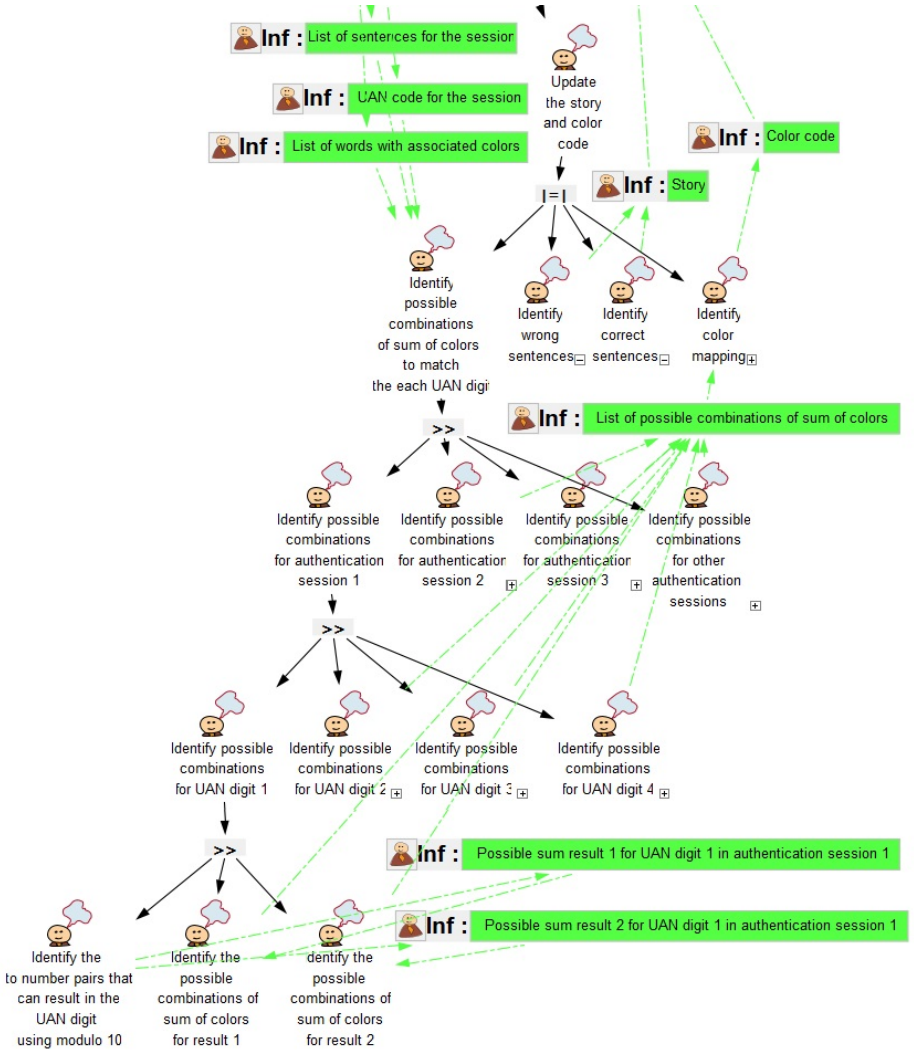


Fig. 6. Extract of the task model of the attacker task "Uncover the long time secret by eavesdropping" for the sub-goal "Update the story and color code"

be less efficient than a textual password, which only requires to remember the code before inputting it (if the user remembers it correctly). There thus may be an efficiency issue for the mechanism. This analysis is consistent with previous studies on the topic. Visual recognition of information has also been used in graphical passwords such as Passface [8]. Usability studies have demonstrated that they outperform passwords and PIN but the that graphical and interaction design might jeopardise their usability [12].

About the security, we analysed that the task is very complex for the attacker. From the task models (Figure 5 and 6), we see that eavesdropping several full sessions are required to uncover the story and color code. Furthermore, the attacker can compromise the attack when trying the story and color code if the temporary guessed story and color code are wrong. From the task model, we also analyzed that there are more than fifty cognitive tasks for trying to find out the story and color code. Moreover, it is very unlikely that an attacker can eavesdrop several complete sessions and complete all of the tasks required to uncover the story and password. To make a comparison with textual password, EEVEHAC is more secure. For a textual password, a single eavesdrop is enough to uncover the password (the task model would be one branch only for the textual password).

To summarize, although the user tasks are numerous, with a potentially high cognitive load that may decrease efficiency, the tasks to attack it are complex. EEVEHAC thus enables to setup a secure communication channel between the user and the target services, in the case where the user needs to use an untrusted computer.

5 Related Work

We did not find any existing approach that explicitly targets usability and security, and that explicitly support the description of attacker tasks. In this section, we present the results of the comparison of the proposed approach with existing model-based approaches to analyse both usability and security, and with existing approaches to take into account attacker tasks.

5.1 Model-based approaches to analyse both usability and security

Braz et al. [6] proposed a so called "Metrics Based-Model", which is a conceptual framework gathering a set of criteria for assessing usability and security. They propose to use it to analyse systematically the usability and security of interactive systems. This approach is based on scenarios of the usage of the system, and it thus cannot cover all the possible user tasks and all of their possible temporal orderings, as scenarios are partial selections of sequences of user tasks. Mona et al. [25] proposed a meta-model of the relationships between usability and security, as well as a knowledge elicitation technique to ensure that required knowledge for the user is taken into account when designing the security mechanism. Our approach also takes into account the user knowledge: procedural knowledge with the hierarchy and temporal ordering of tasks and declarative

knowledge with information to be memorized (e.g. a story). Al Zahrani et al. [1] target healthcare software and proposed a framework, composed of a set of metrics for usability and security and of a fuzzy logic based modeling techniques to produce probabilistic estimations of usability and security of a software. It aims to support decision making for security managers. It takes as input data from security and usability experts. Prior to its application, this approach thus requires a usability and security analysis. Broders et al. [7] proposed a task model based approach focusing on user tasks and on potential threats on these user tasks. In this approach, attack trees are also produced as a mean to ensure that every potential threat on user tasks has been identified. Martinie et al. [21] target the cybersecurity of maritime supply chain and proposed to integrate task modelling to the MITIGATE risk assessment method in order to identify threats on user tasks, whether they root in malicious attacks or in human errors. None of these model-based approaches deal with the attacker tasks.

5.2 Approaches to take into account attacker tasks

Encina et al. [14] proposed a catalog of misuse patterns, that can inform designers and developers when proposing new security mechanisms. This catalog gathers a set of threats and enables to build use cases of attacks. It neither mentions attacker tasks explicitly, nor the attacker in the attack workflow. Ben Othmane [3] et al. proposed to use information about attacker capabilities for risk assessment and mitigation. Such sets of information can be useful but have to be maintained up to date and provide incomplete information about the attacker tasks. Atzeni et al. [2] proposed a method to produce attacker personas, and Moeckel et al. [24] completed the method by adding an attacker taxonomy. Attacker personas aim to raise awareness about threats for the designers and developers, but also more generally in an organisation, to make the security issues more concrete to the users, for example. Personas rarely contain information about tasks, thus they are not sufficient to analyze the complexity of an attack.

6 Conclusion

We presented a modelS-based approach to analyse both user and attacker tasks when designing and developing a security mechanism. The EEVEHAC case study enabled us to show that attack trees are not enough to analyse the security of an authentication mechanism because the attacker tasks for uncovering a key are very complex and this cannot be foreseen from the attack tree. This can only be analysed if attacker tasks are systematically and precisely described. The modeling of attacker tasks enables to identify whether a specific security feature is worth to be implemented. For example, this approach enables to justify that usability could be decreased if there is no other option to reach the required security level. Another example is that if a security mechanism decreases usability but does not make the attacker tasks more complex, it should not be an acceptable design choice. The proposed approach enables to build evidences to argue for design choices.

References

1. Al-Zahrani, F.A.: Evaluating the usable-security of healthcare software through unified technique of fuzzy logic, anp and topsis. IEEE Access **8**, 109905–109916 (2020). https://doi.org/10.1109/ACCESS.2020.3001996
2. Atzeni, A., Cameroni, C., Faily, S., Lyle, J., Flechais, I.: Here's johnny: A methodology for developing attacker personas. In: 2011 Sixth International Conference on Availability, Reliability and Security. pp. 722–727 (2011). https://doi.org/10.1109/ARES.2011.115
3. ben Othmane, L., Ranchal, R., Fernando, R., Bhargava, B., Bodden, E.: Incorporating attacker capabilities in risk estimation and mitigation. Computers Security **51**, 41–61 (2015). https://doi.org/https://doi.org/10.1016/j.cose.2015.03.001, https://www.sciencedirect.com/science/article/pii/S0167404815000334
4. Bernhaupt, R., Martinie, C., Palanque, P., Wallner, G.: A generic visualization approach supporting task-based evaluation of usability and user experience. In: Bernhaupt, R., Ardito, C., Sauer, S. (eds.) Human-Centered Software Engineering. pp. 24–44. Springer International Publishing, Cham (2020)
5. Boldyreva, A., Chen, S., Dupont, P.A., Pointcheval, D.: Human computing for handling strong corruptions in authenticated key exchange. In: Computer Security Foundations Symposium (CSF), 2017 IEEE 30th. pp. 159–175. IEEE (2017)
6. Braz, C., Seffah, A., M'Raihi, D.: Designing a trade-off between usability and security: A metrics based-model. In: Human-Computer Interaction – INTERACT 2007. pp. 114–126. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
7. Broders, N., Martinie, C., Palanque, P., Winckler, M., Halunen, K.: A generic multimodels-based approach for the analysis of usability and security of authentication mechanisms **12481**, 61–83 (2020)
8. Brostoff, S., Sasse, M.: Are passfaces more usable than passwords? a field trial investigation. In: BCS HCI (2000)
9. Carbone, R., Compagna, L., Panichella, A., Ponta, S.E.: Security threat identification and testing. In: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST). pp. 1–8 (2015). https://doi.org/10.1109/ICST.2015.7102630
10. Card, S.K., Moran, T.P., Newell, A.: The model human processor: An engineering model of human performance. In: Handbook of Perception and Human Performance. pp. 1–35 (1986)
11. Daemen, J., Rijmen, V.: The design of Rijndael, vol. 2. Springer (2002)
12. De Angeli, A., Coventry, L., Johnson, G., Coutts, M.: Usability and user authentication: pictorial passwords vs. pin, pp. 240–245. Taylor and Francis Ltd., United Kingdom (Apr 2003). https://doi.org/10.1201/b12800
13. El Batran, K., Dunlop, M.D.: Enhancing KLM (keystroke-level model) to fit touch screen mobile devices. In: Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices and Services. p. 283–286. MobileHCI '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2628363.2628385, https://doi.org/10.1145/2628363.2628385
14. Encina, C.O., Fernandez, E.B., Monge, A.R.: Threat analysis and misuse patterns of federated inter-cloud systems. In: Proceedings of the 19th European Conference on Pattern Languages of Programs. EuroPLoP '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2721956.2721986, https://doi.org/10.1145/2721956.2721986

15. Forte, A.G., Garay, J.A., Jim, T., Vahlis, Y.: EyeDecrypt—private interactions in plain sight. In: International Conference on Security and Cryptography for Networks. pp. 255–276. Springer (2014)
16. Halunen, K., Latvala, O.M.: Review of the use of human senses and capabilities in cryptography. Computer Science Review **39**, 100340 (2021)
17. Hekkala., J., Nikula., S., Latvala., O., Halunen., K.: Involving Humans in the Cryptographic Loop: Introduction and Threat Analysis of EEVE-HAC. In: Proceedings of the 18th International Conference on Security and Cryptography - SECRYPT,. pp. 659–664. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010517806590664
18. Holleis, P., Scherr, M., Broll, G.: A revised mobile KLM for interaction with multiple nfc-tags. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) Human-Computer Interaction – INTERACT 2011. pp. 204–221. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
19. ISO: Iso 9241-11:2018 ergonomics of human-system interaction part 11: Usability: Definitions and concepts. International Organization for Standardization
20. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-hashing for message authentication (1997)
21. Martinie, C., Grigoriadis, C., Kalogeraki, E.M., Kotzanikolaou, P.: Modelling human tasks to enhance threat identification in critical maritime systems. p. 375–380. PCI 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3503823.3503892, https://doi.org/10.1145/3503823.3503892
22. Martinie, C., Palanque, P., Bouzekri, E., Cockburn, A., Canny, A., Barboni, E.: Analysing and demonstrating tool-supported customizable task notations. Proc. ACM Hum.-Comput. Interact. **3**(EICS) (Jun 2019)
23. Martinie, C., Navarre, D., Palanque, P., Fayollas, C.: A generic tool-supported framework for coupling task models and interactive applications. In: Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. p. 244–253. EICS '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2774225.2774845, https://doi.org/10.1145/2774225.2774845
24. Moeckel, C.: From user-centred design to security: Building attacker personas for digital banking. In: Proceedings of the 10th Nordic Conference on Human-Computer Interaction. p. 892–897. NordiCHI '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3240167.3240241, https://doi.org/10.1145/3240167.3240241
25. Mohamed, M.A., Chakraborty, J., Dehlinger, J.: Trading off usability and security in user interface design through mental models. Behaviour & Information Technology **36**(5), 493–516 (2017). https://doi.org/10.1080/0144929X.2016.1262897
26. Naor, M., Shamir, A.: Visual cryptography. In: Advances in Cryptology – EUROCRYPT'94. pp. 1–12. Springer (1995)
27. Nishihara, H., Kawanishi, Y., Souma, D., Yoshida, H.: On validating attack trees with attack effects. In: 39th International Conference on Computer Safety, Reliability and Security, SAFECOMP 2020. Springer
28. Sasse, M.: Computer security: Anatomy of a usability disaster, and a plan for recovery (2003)
29. Schneier, B.: Attack trees (December 1999)
30. Somayaji, A., Mould, D., Brown, C.: Towards narrative authentication: Or, against boring authentication. In: Proceedings of the 2013 New Security Paradigms Workshop. pp. 57–64 (2013)