Int. J. Engng Ed. Vol. 25, No. 4, pp. 745–754, 2009
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Cooperative Visualization of Cryptographic Protocols Using Concept Keyboards*

NELSON BALOIAN[1], WOLFRAM LUTHER[2]
[1] *University of Chile, Dept. of Comp. Cs., Santiago de Chile, Chile. E-mail: nbaloian@dcc.uchile.cl*
[2] *University of Duisburg-Essen, Duisburg, Germany. E-mail: luther@inf.uni-due.de*

Software called CoBo (Cooperative exploring and visualizing cryptographic protocols using concept keyboards), applies the principle of the "concept keyboard" to implement a system that supports the learning of cryptographic protocols. In previous research, concept keyboards (CK) were successfully used to implement a software system supporting the learning of classical algorithms, like QuickSort and Dijkstra's algorithm, among others. Traditional visualization software normally offers the learner an animated representation of the algorithm with the possibility of controlling the execution of the right steps, giving the learner a more or less passive role. Concept keyboards allow the learner to control the execution of the algorithm by deciding which function should be performed when. Four cryptographic protocols were implemented and tested (Wide Mouth Frog, Feige-Fiat-Shamir, Needham-Schroeder and Kerberos V). An initial evaluation confirmed that the use of the CK fosters comprehension of the algorithms, facilitates the learning process and stimulates the learners' activity.

Keywords: concept keyboards; cooperative visualization; algorithm learning; cryptographic protocols

INTRODUCTION

ALGORITHM VISUALIZATION (AV) has often been used in the past to support teaching and learning of complex algorithms and procedures. Many AV systems developed since 1988 are described in three books on *software visualization* edited by John Stasko *et al* [1] and Stephan Diehl *et al* [2], [3]. There are also several repositories of AVs [4], [5] and WWW-pages containing links to collections of algorithm animation. However, most of these systems allow only limited involvement of the learner in the process. In most cases, the system enables the user to control the execution of the algorithm step-by-step, forward and backwards and to define the data set over which the algorithm should be applied.

Some authors have mentioned that increased student involvement in the AV process can lead to better learning results. Hundhausen [6] conducted experiments with students who built their own AV using a prototype language. The system he used for the experiments allowed the specification of the AV in terms of spatial logic, and its interface supported forward and backward execution as well as the dynamic markup and modification of an AV. Hundhausen observed that the experiments allowing students to manipulate the level of involvement showed significantly better results than those allowing them to manipulate the graphical visualization. He concluded that AV software does improve the comprehension level of students and specifically that "what learners do, not what they see, may have the greatest impact on learning" [6].

These observations support our hypothesis that, in order to achieve better comprehension, the student should be able to participate actively in exploring the algorithm.

In a previous work [7], the authors of this paper developed an AV system that applies concept keyboards (CK) to implement a software system supporting the learning of classical algorithms like QuickSort and Dijkstra's algorithm, among others. The idea behind using concept keyboards to support the learning of complex algorithms and protocols is to provide a specialized keyboard containing keys for triggering the execution of certain functions necessary to perform the algorithm. In this way, the learner has a more active role in the execution and visualization of the algorithm, having to decide which function should be performed at what time.

Encouraged by the good results obtained while applying this principle to traditional algorithms and by learners' utterances gathered through several phases of evaluation, we decided to apply it to supporting the learning of cryptographic protocols. Since, in a cryptographic protocol, there is more than one agent involved, the aim of the present work was to develop a collaborative software tool. Based on the hypothesis that collaborative learning activities motivate the learner and lead to better results [8], this tool would help visualize various cryptographic communication protocols using concept keyboards to interactively control execution in a collaborative way. The result

* Accepted 24 February 2009.

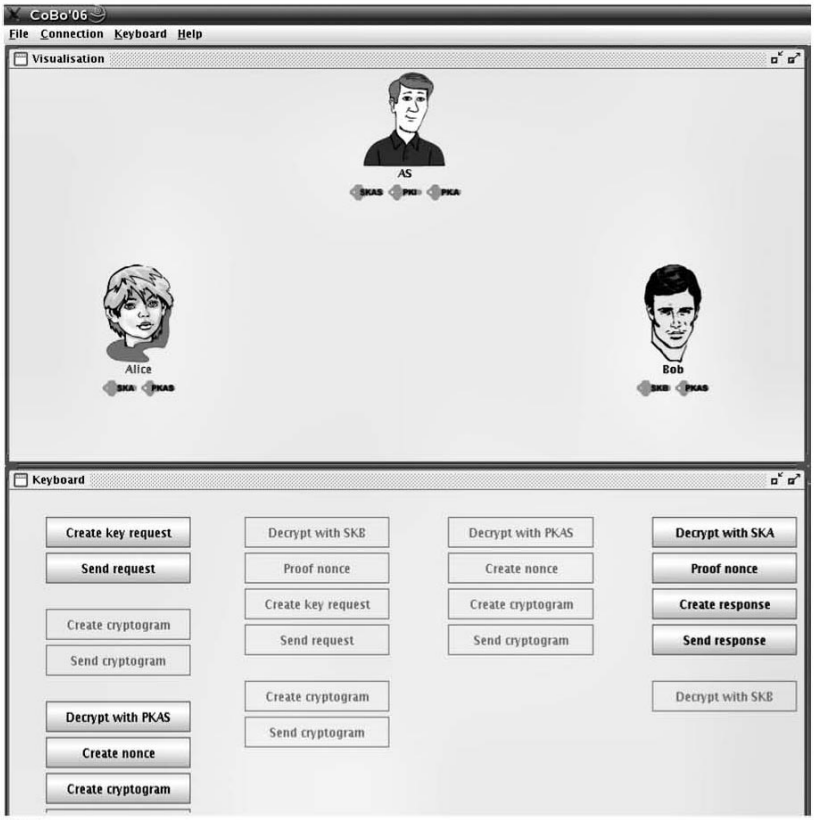

Fig. 1. Screenshot portion of the CoBo software displayed after the Needham-Schroeder Protocol and the "Alice" role have been selected.

was a multi-user software tool that allows users working at different computers to replay the various cryptographic protocols and collaboratively explore, analyze and share the results of their actions.

THE COBO SYSTEM

The CoBo system was conceived in order to implement a Computer Supported Collaborative Learning System based on the visualization and simulation of cryptographic protocols using concept keyboards. This approach has some similarities with the participatory simulation learning technique [9], in which students learn by acting as agents in simulations in which overall patterns emerge from local decisions and information exchanges. The use of a concept keyboard was adopted after the good results obtained when using them for algorithm visualization in the individual learning case.

A learning simulation session with the CoBo system starts with participants connecting to an available server, which will manage the data transfer and coordination between participants. After registering, the participant has to choose a protocol to perform and, after that, the role he/she wants to play in the simulation. Only available (i.e. not yet taken) roles are displayed. After the role has been selected, the main application window is displayed (see Figure 1).

For each cryptographic protocol that the system supports (actually, there are four protocols implemented—the Wide Mouth Frog (WMF), the Feige-Fiat-Shamir (FFS), the Needham-Schroeder, and the Kerberos V protocol), a separate keyboard was implemented that is automatically loaded when the user has chosen the corresponding protocol before the start of a session. An elegant solution using SwixML was employed, which allows describing graphical user interfaces directly in XML for the Java programming language. A SwixML description is interpreted by an engine to create the graphical interface by rendering the graphic objects in an application interface window. This approach was chosen in order to separate the logic of the keyboard design from the program implementing the protocol.

In addition to the concept keyboard, the software displays a window where a graphical animation of the protocol is shown, so that learners can follow the protocol and see the current state of the protocol and the messages that have been already exchanged among the various actors. In this window, the various actors participating in the simulated cryptographic protocol are shown, as are the elements generated during its performance (messages, keys, encrypted messages, etc.) providing a clear visualization of the "knowledge" accu-

mulated at this state of the protocol. This window also shows the animation of the protocol. When a message or a key is created by a user, this element appears as a symbol under the icon representing the user. When a message or key is exchanged between two users, triggered by clicking a corresponding key, the window shows the movement of this element from the icon of the sender to the icon of the receiver.

SYSTEM ARCHITECTURE

The CoBo system is in fact a framework that supports the definition and collaborative simulation of cryptographic protocols. Its architecture has been conceived to incorporate new protocols in an easy way. It is worth noting that an important part of the protocol simulation is inputted into the system as XML files. These files are the Scenario file, the Facts basis file and the Algorithm file. Apart from these files, it is necessary to program the particular functions of the protocol being implemented by extending an existing class called AlgorithmWrapper. It is also necessary to program the classes that will implement the visualization and animation of the algorithm. This architecture implements a strict separation of the code necessary to implement a new protocol from the code that is used to generate the simulation from the XML descriptions. It also allows the re-use of a common communication system by all protocols. This is implemented in a centralized way; that is, all communications between the users is managed by a central server (see Figure 2). In order to use this general mechanism, the data package that will be exchanged between learners should be specified for each protocol by extending an existing class already containing the basic information needed to implement the communication.

As stated above, an important part of the implementation of a new protocol is the definition of the corresponding XML files. The architecture of CoBo was conceived in order to define most of the functionalities of a new protocol with XML files:

The Scenario File. Here, the basic characteristics of the protocol are defined (see Figure 3) including:

- the complete name of the so-called Wrapper-class of the protocol. (Its function will be described in the next chapter.),
- the path for the XML file describing the "facts" of the protocol,
- a short description of the protocol,
- the listing of the actors (roles) involved in the protocol.

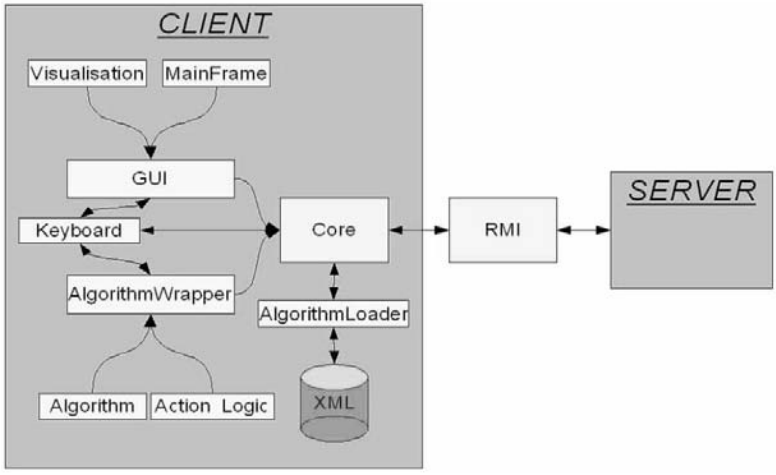

Fig. 2. System architecture showing one client connected to the server [10].

```
<?xml version= "1.0 "?>
<scenario>
   <head>
      <algorithm reflpath="algorithm.needhamSchroederProtocol.NSAsymWrapper"/>
      <xmldata path="algorithm/needhamSchroederProtocol/ns_asym_facts.xml"/>
      <description>
        This is the Needham-Schroeder public key protocol.
      </description>
      <role name="Alice"/>
      <role name="AS"/>
      <role name="Bob"/>
   </head>
   <data>
   </data>
</scenario>
```

Fig. 3. Content of the Scenario XML file for the Needham-Schroeder protocol [11].

```
<?xml version="1.0" encoding="utf-8"?>
<content language="englisch" date=05.07.2007">
   <keyboard>
      <panel Layout="null">
         <Button id="1" Bounds="35,30,160,30" type="Alice"
          Text="Create key request"
          HorizontalTextPosition="CENTER" VerticalTextPosition="BOTTOM"
          tooltiptext="Alice initiates the communication through a request to AS
          in order to get Bob's public key." ActionCommand="1" />
          .......
      </panel>
   </keyboard>
   <methodmatch>
      <method name="createMessage1ForAS" id="1"/>
         .......
   </methodmatch>
   <description>
      <!--     Protocol description     -->
   </description>
   <alogic type= "2">
      <!--     Correspondig Petri net     -->
   </alogic>
</content>
```

Fig. 4. Excerpts of the content of the Fact basis XML file for the Needham-Schroeder protocol [11].

The facts basis and protocol description file

This file is divided in four parts, each describing a fundamental part of the protocol (see figure 4):

- Keyboard Definition (<keyboard>). This part describes the graphical layout of the concept keyboard that will be used to control the simulation of the protocol. The size, position, and labeling of each key are defined here. It is also possible to input help text and associate with certain keys. Using the "type" attribute, each key is associated with a certain role. The action triggered by each key is defined by a number given to the ActionCommand attribute (see following point).
- Method Mapping (<methodmatch>). Here, the methods of the protocol programmed in the AlgorithmWrapper class are assigned to a key. Each method of the wrapper class is numbered, and this number is associated with a key.
- Protocol description (<description>). This block contains an HTML description of the protocol. This description will appear in the Description sub-window of the user interface (upper right in the screenshot in Figure 1 above). This description includes the stages of the protocol, the actors involved and the symbols that appear in the protocol visualization window.
- Action logic (<alogic>). The last part of the Fact basis file is the XML description of the protocol. The generation of this description is explained in the next section.

ACTION LOGIC GENERATION FOR PROTOCOL DESCRIPTION

Each cryptographic protocol has a unique correct chronological sequence of actions that is defined in the action logic. The action logic is implemented with a kind of state machine that controls the correctness of all the actions taken by the participants. When a correct action is executed, the machine reaches the next state, which in turn requires a new action to be launched by the actor. Otherwise, an error message is displayed in order to help the user find the right action. If the next proposition is also wrong, the resulting error message is more concrete. If the user makes a third mistake, the right action is proposed.

There are several options for implementing the action logic. We first tried using a stack machine and then a finite state automaton; finally, we decided to use an enhanced PetriStateMachine (PSM), which interprets a Petri net with so-called message and error places. The PSM checks the states (nodes) and notices each modification of the number of tokens. Furthermore, there are layered error messages attached to the nodes that are displayed in a certain order to support the user (see Figure 5).

Another extension allows an executable program to be launched as part of a Petri net. When these transitions are used, precise rules and heuristics should be formulated in order to limit the possibility of these transitions being fired. A text-executable transition was used in the WMF protocol to restart the protocol when a participant rejects a transmitted key. Further details are reported in [10].

The Petri net has a starting place ("Start Node") containing one token at the beginning followed by two transactions concerning the first action (createMessage1ForAS), which creates a message for the authenticity server; alternatively, the transition "!s*sonst" handles the error case (cf. Figure 5). The complete Petri net can be found in [11]. The Petri nets as well as the error messages are generated as XML files using a powerful graphical editor called Freestyler [12]. The XML files are part of the Facts basis of each protocol.

IMPLEMENTED PROTOCOLS

WMF: Wide Mouth Frog is a computer network authentication protocol designed to be used on insecure networks. It allows individuals

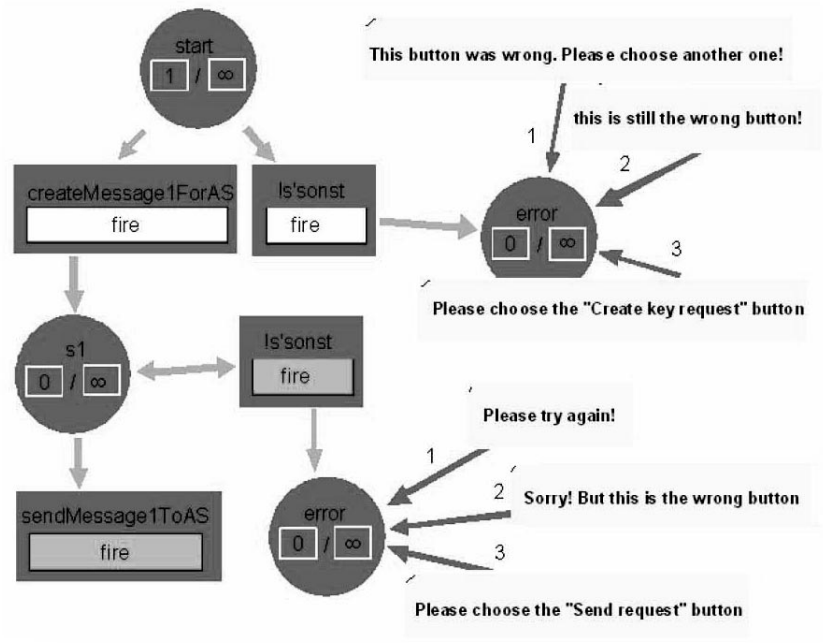

Fig. 5. Part of the Petri net for the Needham-Schroeder protocol [11].

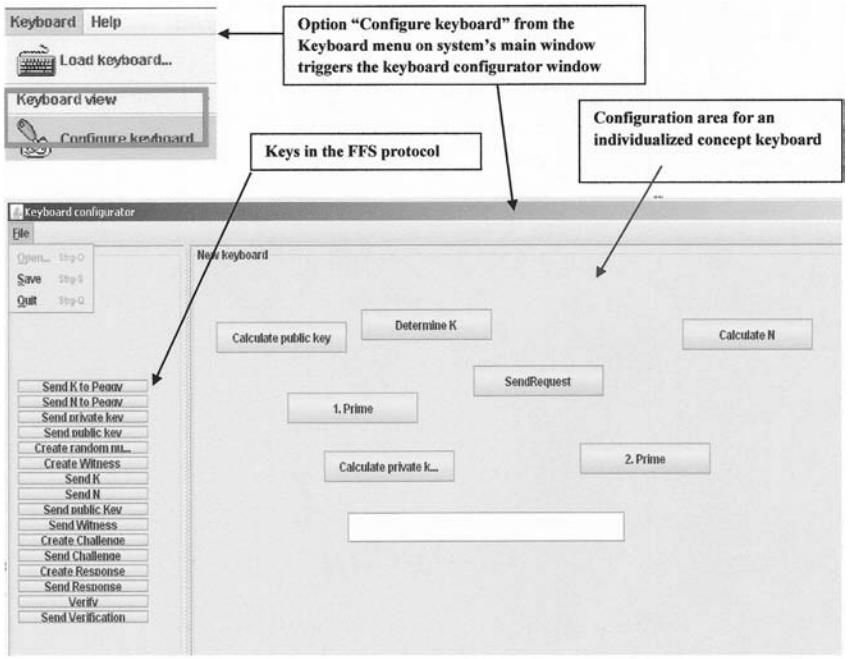

Fig. 6. Configuration dialogue for an individualized keyboard [24].

to communicate over a network to mutually authenticate their identity while, at the same time, preventing eavesdropping or replay attacks. It provides mechanisms for detection of information modification and the prevention of unauthorized reading.

FFS: The Feige-Fiat-Shamir identification scheme, introduced by its authors in 1987, is based on a zero-knowledge protocol. Alice proves to Bob that she knows secret numbers without revealing any information about her secret. The protocol uses public-private keys requiring only a few modular operations in a parallel verification process. It is fast and can be applied in smart card applications. FFS solves the identification and signature problem but cannot be used for data encryption.

Kerberos is a computer network authentication protocol. It is designed to provide strong authentication for client/server applications and was

created by MIT as a solution to well-known network security problems. The Kerberos protocol uses secret-key cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to ensure privacy and data integrity (http://web.mit.edu/Kerberos/).

The Needham-Schroeder Public-Key Protocol is a computer network authentication protocol designed for use across insecure networks. Our simulation uses a fixed version of the protocol, in which the communication was slightly altered in order to prevent the possibility of a man-in-the-middle attack.

FURTHER SYSTEM FEATURES

The system can be set up by a user in "teacher" or "learner" mode. In teacher mode, the user can assign roles to the different users or combine all the roles in one person. In this mode it is also possible to monitor the execution of the protocol and intervene if necessary in order to guide the learners. In learner mode, the user is assigned a certain role in the protocol and receives a keyboard containing keys for the actions he is supposed to trigger.

The system also offers learners a help functionality in case they are unable to continue the execution of the protocol. The assistance consists of HTML documents containing a detailed overview of each protocol and an explanation of the operations (the steps) of the relevant algorithm. The users can search for keywords contained in the hierarchically organized help files.

In addition to the help functionality, there is a quick guide containing a shortened description of the protocols used in the implementation, as well as the icons that are used to represent the various actors and elements involved in the protocol. This enables the learner to recognize the symbols appearing in the software's interface and their meaning. In order to help the learners synchronize their actions, the system also provides a chat tool.

Learners also have the option of configuring their own concept keyboards, starting with the one proposed by the system and taking the role of a designer deciding the outline of the concept keyboard (see Figure 6 above). This functionality provides learners with a deeper understanding of the protocol—which actions are related, which and what their roles are. The new keyboard can be configured by "drag and drop" of the "keys" ("buttons") to the desired location on the virtual keyboard surface.

The system has been implemented for desktop PCs as well as mobile devices (PDAs). Mobile devices allow users to synchronize their actions through face-to-face communication when the system is used in a classroom scenario. This was because some authors have reported that mobile computing can help to foster social interaction among students, raises the motivation and promotes discussion and reflection [13].

RELATED WORK

In the field of visualization of algorithms in general, the following works were of particular interest to us: COMET, which was developed by Feiner and McKeown [14], is an experimental testbed for the interactive or automated generation of multimedia explanations that combine text and three-dimensional graphics on the fly. In response to a user request for an explanation, COMET decides which information should be expressed in which medium.

The project WIP (Knowledge-Based Presentation of Information, www.dfki.de/imedia/wip/) seeks to develop a presentation system that is able to generate a variety of multimedia documents and to present the same information in different ways, depending on the generation parameters and the individual users in particular communicative situations.

Our work was also inspired by M. Eisenberg [15], who offers a number of interface guidelines for mathematical algorithms and suggests ways to provide the user with flexible means for both controlling and understanding the algorithm in question.

In Bridgeman et al. [16], a platform-independent e-learning tool, PILOT, was designed; it allows for the generation of random instances of a problem, user interaction specifying a solution, evaluation of solutions and generation of correct solutions to the problem. Our work has points in common with initiatives concerning XML user interface languages like the eXtensible Interface Markup Language (XIML, see http://www.ximl. org/) to configure interfaces. The recently presented Matrix-Pro system [17] allows the instructor to interact with any data structure already implemented in a library. Its framework allows users to create animation sequences and to combine them seamlessly. The main view of the program consists of a menu bar, a toolbar and a visualization area. Recent works, such as the frameworks Ganimal [2] and LEONARDO [4], use an event-driven approach to specify the relevant events associated with an algorithm or data structure.

In the field of security algorithms and protocols visualization the following works are relevant.

CyberCIEGE (http://cisr.nps.navy.mil/cyberciege/index.htm) is a system that allows the creation of a virtual world in order to teach and learn security concepts. It was conceived as an interactive environment, like a video game, that covers significant aspects of network administration and its security. The learners have to configure operative systems, servers and applications, taking into

account both the security and the performance of the system.

ProtoViz (http://www.cs.chalmers.se/_elm/courses/security/) is an application implemented as a java applet developed in 2003. It allows the visualization of cryptographic algorithms defined by the user though a special grammar.

GRASP (GRaphical Aid for Security Protocols) [18] is a tool for teaching security protocols by means of animations. It was designed by the United States Air Force Academy to be used in lectures on computer security. Its purpose is to graphically show the different protocols and allow the learner to introduce variations and visualize the security changes they produce.

GRACE [19] is an educational tool that supports learning of cryptographic protocols. It is also conceived to adopt an active learning model that engages the learner with a request to describe, in an exemplification of a real-world scenario, cryptographic protocols using simple primitives whose effects are visualized by means of animated sequences. It offers several cryptographic and non-cryptographic related operations with their respective visualizations. By executing a series of these operations in the proper order, a teacher is able to provide a visual introductory description of several protocols. Cryptographic operations are not just simulated but implemented.

According to the revised literature, although the previous works allow a fair level of involvement on the part of the learner in the control of the algorithm execution, no one uses a collaborative learning approach.

COLLABORATIVE LEARNING (CL) WITH COBO

Many authors back the idea that collaboration learning activities enriches the learning experience ([20], [21], and [22]). Some even say that learning is only possible in a social context [23], since learning occurs by contrasting one's own knowledge with the knowledge of the rest. There has been abundant research on the factors that make for effective CL which can be summarized in the following five:

- **Individual responsibility.** All members of the group are responsible for their own work, role, and efforts to learn. In the CoBo environment this is achieved by giving the students individual roles with clear rules they have to follow.
- **Mutual support.** In addition to being responsible for their own learning, each member is responsible for teaching other members of the group. In the CoBo scenario, participants who know the protocol can explain to those members of the same group who do not, by telling them when and why they have to trigger a certain action.
- **Positive interdependence.** Achieving the personal goal of each participant depends on achieving the group goal. This is naturally present in the CoBo scenario since every participant depends on the actions of the rest of the group to achieve a personal goal, which is to complete the protocol.
- **Social interaction**. Decision-making must involve discussion among collaborators. CoBo includes communication mechanisms to facilitate the discussion in order to decide collaboratively which step should be the next in the protocol. The mobile version of CoBo was developed to facilitate face-to-face discussion and social interaction.
- **Formation of small groups.** Communication, discussion, and consensus building can only be achieved in small groups. As cryptographic protocols involve 2–3 persons, we will have only small groups, where discussions and consensus can be achieved.

As can be seen from the previous analysis, CoBo implements a promising collaborative learning environment since it meets the principal requirements for effective CL.

The principal motivation we had to develop such a scenario was that collaborative learning has been shown to be more motivating and to have positive impacts in other social aspects, like developing group skills (leadership, resolution of conflicts, negotiation).

EVALUATION OF RESULTS

A group of forty computer science students (aged 20 to 35, average age 24.28) was randomly chosen from a third-year course on cryptography and network security at the University of Duisburg-Essen. They worked with the Wide Mouth Frog, the Feige-Fiat-Shamir and the Kerberos protocols implemented for desktop computers. The students had just been introduced to classical communication protocols (approximately 75% assistance) and had never used this kind of visualization software before. In a pretest, the majority estimated their own knowledge as low (WMF 50%, FFS 57.5%, Kerberos 52.5%).

The evaluation was carried out by S. Selvanadurajan in two parallel sessions in 2008 from May 14 to June 8, and the results were reported in her master's thesis [25]. First, the participants were asked to evaluate the software in the teacher role and to fill out a questionnaire with 45 items allocated to ten hypotheses. Then, 13 groups of three persons were formed to explore the algorithms in a collaborative way and to answer 18 questions, corresponding to seven hypotheses.

The first part of the evaluation concerned ten hypotheses associated with 45 statements to be judged by signaling agreement or disagreement (B2) or by using the five-point Likert scale (L5). Low values 1 and 2 indicated (strong) rejection (–), and high values 4 and 5 (strong) agreement (++); 3 indicated indecision.

Four hypotheses were aimed at testing whether

working with the concept keyboard facilitates understanding of cryptographic protocols. These are explained in the following list. For each hypothesis, the number of related questions is given, as is the type of evaluation used (L5 for the Likert scale and B2 for agreement or disagreement). For those hypotheses where the Likert scale was used to answer the corresponding questions, the median value, the range, the mean value and the variance are also shown. For those hypotheses where agreement or disagreement was used to answer the related questions, the level of agreement is shown as a percentage.

- Use of the CK supports the learning the entire extent of cryptographic protocols. (2, L5, Median 4, Range 1–5), Mean = 4.01, Variance = 0.89
- The free exploration and the possibility of taking erroneous actions foster the learning process. (4, B2, 75%)
- The CK makes the learning process more efficient. (4, B2, 97.5%)
- Interacting with the CK stimulates the motivation. (3, L5, Median 4, Range 1–4, 74%), Mean = 3.93, Variance = 0.97.

The remaining hypotheses dealt with the interface design of the keyboard and the output device:

- The keyboard is well structured. (1, B2, 95%)
- The error messages are comprehensible. (4, B2, 82.5%)
- The help desk contains all necessary information to support the user during the learning process. (5, B2, 85%)
- Concise error information and explanations in response to specific questions enable the user to learn the protocols. (2, L5, Median 4, Range 1–5, 70%) Mean = 3.925, Variance = 0.938
- The screen design assists the understanding of the protocol step-by-step. (17, B2, 94%)
- The visualization does not create obstacles for the interaction. (3, B2, 92.5%).

The hypotheses used in the group evaluation had a supplementary character and were directed at evaluating the collaborative learning.

- The group-based learning provides only a partial view of the protocols to the individual actor. (1, L5, Media 0, Range 1–5, 48%), Mean = 2.79, Variance = 1.51
- It is possible to allocate the actions to the actors that triggered them by interpreting the visual output. (2, B2, 85%)
- The customized and individualized keyboard promotes deeper understanding of the protocol. (4, B2, 82.5%)
- The use of individualized concept keyboards in the cooperative scenario helps to learn the protocols in an efficient way (3, B2, 86%)
- To learn cryptographic protocols in a group with individualized keyboards is more motivating. (3, L5, Median 4, Range 1–5, 63%) Mean = 3.68, Variance = 1.25.

The following two additional hypotheses were aimed at testing design aspects of the human-machine interface.

- The individualized concept keyboard is well arranged. (2, B2, 70%)
- The inbuilt chat functionality displays error messages and gives feedback from the other pupils. (3, B2, 88%).

The questionnaires were evaluated using several tools. The test persons' data (name, age, studies, etc.) and the answers were transferred to an Excel spreadsheet. The answers were analyzed with the statistics software SPSS, displaying various kinds of diagrams and applying appropriate hypothesis tests, like the Mann-Whitney U-test.

At the end of both tests, students were asked whether they preferred to take the teacher's role or one of the protocol actors' roles. It was interesting to notice that 75% voted for the teacher perspective arguing that this role:

- allows for better exploration of the protocols,
- gives more time to consult the help screens without the time constraints that occur in a collaborative scenario,
- it shows the complete keyboard with all necessary actions in the right order in the start-up configuration, thus leading to deeper understanding when using keys in a different order.

As an advantage of the group version, the participants mentioned that the various actors and their roles were clearly discernible and that people could help each other by using the chat or communicating directly face-to-face.

We wanted to find out whether the participants performed better in a post-test organized afterwards than in a pre-test before working with the CK. In both tests, students were asked to perform the protocols' actions in the right order. The non-parametric Mann-Whitney U-test was used to examine the collected answers. This test makes no assumption about the distribution of data. The hypotheses for the comparison were:

- H_o: The two samples come from identical populations; the results of both tests are indistinguishable.
- H_1: The two samples come from different populations.

We transformed the samples into a ranked list, eliminating the entries of students that solved the problems without errors in both cases. Then we calculated the test variable U and the ratio $z = (U-\mu_U)/\sigma_U$. This value is compared to a table of critical values for z based on the sample size of each group.

If z exceeds the critical value at a significance level (usually 0.05), it means that there is evidence for rejecting the null hypothesis in favor of the alternative hypothesis. Here, the calculated z is compared to the standard normal significance levels.

We found that the value of z was 2.65 > 1.96 and the hypothesis H_o had to be rejected. Therefore, on a 95% level of significance, people performed better in the post-test.

In an additional study we contrasted similar statements from the individual and the group test to examine whether the participants in the single test scenario performed significantly differently from the ones in the group scenario.

The first item was related to the interaction with the CK and the visualization of the actions launched by the user. Further questions dealt with the CK design and the ability to hit erroneous keys. For each of these statements, the Mann-Whitney test showed no difference at a 95% significance level.

There were six further questions which were not allocated to any hypothesis. They were asked in order to learn more about the contribution of the individual configuration of the concept keyboard to the correct performance of the algorithm. Of the people interviewed, 95% agreed that the configuration of the keyboard by drag and drop is quite simple and that they tried to place the keys in a semantically meaningful way, for example, keys in the right order, encryption on the left and decryption on the right.

The contribution of adding sounds or special graphic information to the keys was declined (70% disagreement), but individual shapes of the keys were welcomed by 75% of the participants.

Further information was obtained in the interviews concerning form, positioning and captions of the keys. Following these suggestions, the keyboards in the teacher role were redesigned in some cases. Each key now has an intuitive tool tip explaining its functionality. Also, there are now more precise error messages displayed in red in the chat window.

To become familiar with CoBo, a new quick guide was created that quickly refreshes the protocols, explains the features of the software and gives more emphasis to the configuration of the keyboard by the users. In a later version described in a master's thesis by A. Kováčová [11], the Needham-Schroeder protocol was added using symmetric and asymmetric encryption and allowing users to simulate the man-in-the-middle and the reply attack against this protocol.

This evaluation clearly shows that the use of the CK fosters the comprehension of the implemented communication protocols. The main point seems to be that users can choose which actions to take. This can be done proceeding step-by-step when the keys are organized in the right order on the keyboard or, alternatively, via a self-configured keyboard that allows students to learn from their mistakes. Furthermore, two different scenarios were implemented: a teacher's role for an individual and a group version with different keyboards mirroring the roles of different actors within the protocol. Both exploration modes of the CK approach facilitate the learning process and motivate the user to study the protocols with the CoBo software.

Several hypotheses concerning design aspects of the keyboards as well as the visualization and animation output met with high approval ratings. Participants suggested increasing the number of communication protocols presented and providing an extended explanation of the protocols being studied. This has been implemented in a new version of the CoBo Software.

CONCLUSIONS

We have presented a new way of learning interactive cryptographic protocols by using concept keyboards and sophisticated visualization and animation features. In this approach, learners are asked to configure their own steering interfaces and to select actions to be performed in a correct order.

The framework developed helps to build complete protocol simulation software by using software design patterns, prefabricated parts of the action logic realized in a colored Petri net, an XML-based interface rendering language and standard file formats to describe the facts of the protocols. The software architecture, realized for the first time within a tool that supports the animation of standard algorithms [7], implements a new controller design that allows users to select a class library on the fly, to instantiate objects and to call objects. Individual input and output interfaces are configured through XML files.

One of the contributions of this work is to explore the possibilities of merging collaborative learning with algorithm visualization. As future work we envisage the testing of the collaborative learning scenario against the individual learning one, taking into account that collaborative learning activities are not only adopted in order to improve the learning of the subject but it also has other positive "side effects" [21].

Acknowledgements—we thank our master's thesis students for their valuable contributions. This work was partially funded by FONDECYT grant number 1060797.

REFERENCES

1. J. T. Stasko et al., *Software Visualization—Programming as a Multimedia Experience*, MIT Press (1998).
2. St. Diehl, C. Görk, and A. Kerren, Preserving the Mental Map Using Foresighted Layout, *Proc. Eurographics—IEEE TCVG Symposium on Visualization*, May, 2001, Ascona, Switzerland, (2001).
3. S. Diehl (ed.), *Software Visualization, State-of-the-Art Survey*, LNCS 2269, Springer (2002).

4. P. Crescenzi, N. Faltin, R. Fleischer, Ch. Hundhausen, St. Näher, G. Rössling, J. Stasko, and E. Sutinen, The Algorithm Animation Repository, *Proceedings of the Second International Program Visualization Workshop*, Århus, Denmark (2002) pp. 14–16.
5. G. Rößling, *Kommentierte Sammlung von Algorithmenanimationen und Animationssystemen*. http://www.animal.ahrgr.de/Anims/animations.php3 (Last visited August, 2008).
6. C. D. Hundhausen, *Toward effective algorithm visualization artifacts: Designing for participation and communication in an undergraduate algorithms course*, unpublished Ph.D. dissertation, Department of Computer and Information Science, University of Oregon (1999).
7. N. Baloian, H. Breuer, W. Luther, Concept keyboards in the animation of standard algorithms, *J. Visual Language and Computing*, **19,** (2008), pp. 652–674.
8. C. Collazos, L. A. Guerrero, J. A. Pino, S. F. Ochoa, Evaluating Collaborative Learning Processes. Lecture Notes in Computer Science **2440,** (2002), pp. 203–221.
9. V. Colella, Participatory simulations: Building collaborative understanding through immersive dynamic modeling, *J. the Learning Sciences* **9, (**2000), pp. 471–500.
10. B. Weyers, *Concept Keyboards zur Steuerung und Visualisierung interaktiver kryptographischer Protokolle CoBo'06*, University of Duisburg-Essen (2006).
11. A. Kováčová, *Implementierung des Needham-Schroeder Protokolls in einer verteilten Simulationsumgebung für kryptografische Standardverfahren,* Master's thesis, U. of Duisburg-Essen (2007).
12. COLLIDE, Universität Duisburg-Essen. Freestyler. http://www.collide.info
13. T. Liu, H. Wang, T. Liang, T. Chan, W. Ko and J. Yang, Wireless and mobile technologies to enhance teaching and learning, Journal of Computer Assisted Learning **19**(3), (2003), pp. 371–382.
14. S. Feiner, K. McKeown, Automating the generation of coordinated multimedia explanations, *IEEE Computer* **24**(10), (1991), pp. 33–41.
15. M. Eisenberg, The thin glass line: Designing interfaces to algorithms, in: Proceedings of the *SIGCHI Conference on Human Factors in Computing Systems: Common ground*, ACM Press (1996), pp. 181–188.
16. S. Bridgeman, M. T. Goodrich, S. G. Kobourov, R. Tamassia, PILOT: An interactive tool for learning and grading, Pro*ceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, ACM Press (2000), pp. 139–143.
17. V. Karavirta, A. Korhonen, L. Malmi, K. Stalnacke, MatrixPro—a tool for on-the-fly demonstration of data structures and algorithms, *Proceedings of the IEEE International Conference on Advanced Learning Technologies* (ICALT'04), IEEE (2004), pp. 26–33.
18. D. Schweitzer, L. Baird, M. Collins, M.Sherman, GRASP: A visualization tool for teaching security protocols, *Proc. of the 10th Colloquium for Information Systems Security Education*, June (2006).
19. G. Cattaneo, A. de Santis, U. Ferraro Petrillo, Visualization of Cryptographic Algorithms with GRACE, *J. Visual Languages and Computing* **19**(2), (2008), pp. 258–290.
20. P. Dillenburg (Ed.), *Collaborative Learning: cognitive and computational approaches,* Pergamon, Elsevier Science Ltd. Oxford, England (1999).
21. D. Jhonson and R. Jhonson, *Learning together and alone,* Prentice Hall. New Jersey. USA (1999).
22. N. Webb, G. Baxter, L. Thompson, Teachers grouping practices in fifth grade science Classrooms, *The Elementary School Journal* **98**(2), (1997), pp. 91–113.
23. L. S. Vygotsky, *Mind in Society: The development of higher psychological processes*, Harvard University Press. Cambridge, MA, USA (1978).
24. L. Selvanadurajan, *Interaktive Visualisierung kryptographischer Protokolle mit Concept Keyboards—Testszenarien und Evaluation,* Master's thesis, University of Duisburg-Essen (2007).

Nelson Baloian is an associate professor in the Computer Science Department of the University of Chile in Santiago de Chile. His main research topics are computer-supported collaborative learning and distributed systems. He is a visiting lecturer at the University of Waseda and the University of Duisburg-Essen.

Wolfram Luther is a full professor of computer science at the University of Duisburg-Essen. He leads a research group of a dozen persons in scientific computing, computer graphics and image and text processing. The team specializes in the development of software and algorithms with result verification and of interactive teaching and learning systems in several contexts.