DISCERN: A Collaborative Visualization System for Learning Cryptographic Protocols

Giuseppe Cattaneo
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, Via Ponte don Melillo, I-84084 Fisciano (SA), Italy
Email: cattaneo@dia.unisa.it

Alfredo De Santis
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, Via Ponte don Melillo, I-84084 Fisciano (SA), Italy
Email: ads@dia.unisa.it

Umberto Ferraro Petrillo
Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma "La Sapienza", P.le Aldo Moro 5, 00185 Rome, Italy.
Email: umberto.ferraro@uniroma1.it

***Abstract*—In this paper we propose a novel approach to the learning of cryptographic protocols, based on a collaborative role-based visualization system, DISCERN, that helps students to understand a protocol by actively engaging them in a simulation of its execution. In DISCERN, each student shares a visual exemplification of a real-world scenario with other students and impersonates one of the parties involved in the execution of a protocol. Students may take the role of legal or malicious parties and are provided with primitives that are useful for the implementation of several protocols. To achieve a certain security goal correctly, legal parties have to collaborate and carefully execute the steps required by the implemented protocol in the correct order. If any error is made, the security of the protocol is exposed to the threats coming from other students impersonating malicious parties. The entire process is run under the supervision of the teacher.**

I. Introduction

A cryptographic protocol is a distributed algorithm defined by a sequence of steps indicating precisely the actions required of two or more entities (also called, parties) to achieve a specific security objective ([1]). The steps of a cryptographic protocol consist of computational operations or message transmissions that must be executed in a specific order by the parties involved in the protocol. The objective of a cryptographic protocol is usually the transmission of a message under certain security conditions such as *integrity* (ensuring information has not been altered by unauthorized or unknown means), *confidentiality* (keeping information secret from all but those who are authorized to see it), *authentication* (corroboration of the identity of the entities involved in the protocol) and *message authentication* (corroborating the source of information).

Understanding cryptographic protocols may sometimes be difficult due to the numerous interactions that may occur between the parties of a protocol and because of the complexity of the mathematics behind the computation operations. For these reasons, cryptographic protocols are often taught with the help of visual metaphors. The teacher draws a scenario, typically using electronic slides, where several parties are engaged in the execution of a cryptographic protocol. The protocol specification is provided by simulating its execution and visualizing, at each step, the computation being executed or the message being transmitted. Moreover, the protocol execution is often simulated using a real input set, albeit a very simple one, to show the learners the effects of the mathematical computations used by the protocol.

This approach is very simple to implement and leads to lectures that tend to be easier to understand than traditional ones. However, it suffers from the limitations of a passive style of teaching. Several studies in the past, such as [2]–[4], have proved that involving computer science students in an active learning environment improves their comprehension and retention of materials. When speaking of cryptographic protocols, an interesting form of involvement for the students would be the possibility of letting them see for themselves the behavior and (some of) the security properties of a cryptographic protocol by executing one. To this end, several tools have been proposed so far to simulate, through visualization, the execution of a cryptographic protocol. All of these tools share essentially the same philosophy: they require a person to interact with an application to dictate/describe all the steps that form a certain cryptographic protocol: this implies that the student using one of these tools gets, as output, exactly what he instructs the tool to do.

In this paper, we propose a different approach by making use of the distributed nature of a cryptographic protocol. In our approach, several users impersonate different parties involved in the simultaneous execution of a protocol. They can collaborate in the execution of the protocol, in order to achieve a specific security objective, or they can hinder the protocol execution, in order to disrupt its security. Our expectation is that by properly playing his part during a protocol execution involving other students and with a clear objective to be reached under the supervision of the teacher, a student is better motivated in experimenting with a cryptographic protocol and is able to reache a stronger comprehension of the protocol than when using a traditional single-user visualization system,

We implemented this approach as a java-based distributed system named DISCERN. Our system uses the same visualization and cryptographic engine as GRACE, an existing visualization tool, and adds all the complexity needed to

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360

handle multi-user protocol execution in a scalable, consistent, and efficient way. A prototype version of DISCERN can be found at:
`http://www.dia.unisa.it/research/discern/`.

II. Previous Work

Over the past few years, several tools have been proposed to support and simplify the teaching of cryptographic protocols through software visualization. These tools can produce a visual, sometimes interactive, description of a protocol, thus replacing traditional electronic slides. The resulting visualizations typically depict a scenario where several parties coexist and interact to execute a protocol. For example, the ProtoViz [5] system accepts as input a protocol description written in a simple specification language and outputs a visualization of the protocol. The visualization employs graphical metaphors commonly used in this field, such as icons displaying a key to represent a cryptographic key, and is animated in such a way as to represent the stepwise actions of the protocol.

TECP [6] and GRASP [7] place stronger emphasis on the mathematics underlying a protocol, by explicitly visualizing the computational steps and the information exchanged during a protocol execution without using animations or graphical metaphors. Moreover, these tools introduce the concept of *interactivity* by making it possible for the user to modify on the fly the specification of a protocol to recreate, for example, what-if scenarios.

Another interactive visualization system is GRACE [8], whose main distinction with respect to other systems is that the cryptographic primitives it uses are concretely implemented. Running a protocol on real input sets makes it possible to experience firsthand some of its properties and weaknesses. The drawback of this system is that a protocol can be represented only if the cryptographic primitives it uses have been previously implemented in the system. Finally, JCrypTool [9] is an interactive e-learning application that can be used to analyze the behavior of several cryptographic algorithms and protocols. Like GRACE, the cryptographic primitives available in JCrypTool are concretely implemented and may be run on arbitrary data sets. On the other hand, this system makes litte use of visual metaphors to explain cryptographic protocol related concepts and is more focused on the numerical part of these protocols.

To the best of our knowledge, the collaborative approach has not yet been tried in the field of software systems for learning cryptographic protocols. The only collaborative approach we are aware of has instead been proposed in [10] where the author describes a paper-and-pencil game that can be used to simulate cryptographic protocols and to explore possible attacks against them. The game is played by several users at a time, each playing a role in the simulated protocol. By following a provided set of rules that mimic the cryptographic operations used by a protocol and by interacting with each other, the learners can recreate a protocol and deal with some of the key issues in cryptographic protocols, such as integrity, confidentiality, and non-repudiation in secure data communications.

III. Our Contribution

All the interactive visualization tools presented so far require a single user to simulate the steps that several parties have to run for the correct (or incorrect) execution of a protocol. This approach works well with teachers, i.e., when the focus is only to present a protocol and its properties, and one clearly has in mind the steps that have to be executed. This is often not the case when students are interested in learning or improving their comprehension of a protocol. In such cases, the user has typically a very limited knowledge of a protocol and its learning would benefit from the possibility to experience the security properties of that protocol and to assess the consequences of errors made during the simulation of the protocol itself.

The process of learning a protocol by experimenting with it may be difficult to follow if the user is asked to impersonate, simultaneously, all the parties involved in a protocol execution, including those threatening the security of the protocol. We thus make recourse to the original nature of a cryptographic protocol: a distributed algorithm where several parties have to cooperate to achieve a security goal. If all the parties behave correctly, the security goal will be achieved. If one of the parties deviates from the standard protocol (e.g., by executing two consecutive steps in the wrong order), the protocol execution may be exposed to security threats.

Our idea, implemented by a distributed system called DISCERN, is to involve several students simultaneously in the simulation of a cryptographic protocol, with each user impersonating a distinct party. All parties share a visual exemplification of a real-world scenario in which they are connected via a public communication network, can instantiate and manipulate simple artifacts (e.g., a digital document) and can perform a set of communication and cryptographic primitives related to these artifacts. By using these primitives, parties can recreate several cryptographic protocols and some of their application scenarios using real input sets. Moreover, these primitives are not only simulated but concretely implemented: they will behave in the simulation as they would behave in a real-world implementation. A key factor is that some of the connected users, even the teacher himself, may behave dishonestly and may use some ad-hoc primitives (such as eavesdropping on the communication channel or forging digital credentials) to threaten the security of the protocol. Thus, the student is engaged in an activity that may be either *collaborative* or *competitive*. It is collaborative because legal (i.e., honest) parties have to to coordinate and cooperate with each other to achieve a security objective using a cryptographic protocol. It is competitive because legal parties are interested in running correctly a protocol in order to prevent malicious (i.e., dishonest) parties from accessing a secret information and, on the other side, malicious parties are interested in using all the primitives they have and all the errors made by legal

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360

parties to threaten the security of their protocols and access their secret information.

IV. Application Scenarios

In this section we describe three learning activities, lasting approximately 10-15 minutes each, where topics related to cryptographic protocols can be tried out by a small group of students using DISCERN. We will suppose that the students involved in this activity already have some basic theoretical knowledge of public-key related cryptographic protocols and introduce three application scenarios that require the implementation of these protocols. The purpose of this section is to provide, by means of several examples, a description of the possible way to use DISCERN for teaching cryptographic protocols and to give some insights on the possible advantages coming from the adoption of our approach.

For each scenario, we provide a short description of a security problem to be solved using a cryptographic protocol. We introduce an initial setting for the session and illustrate a possible protocol that apparently solves that problem. Moreover, we outline some of the possible threats to the protocol due to errors performed by legal parties and/or to actions undertaken by malicious parties. Finally, we summarize the concepts that we expect students may better understand through such an experience.

A. Secure Communication using Public-Key Cryptography

In a public key setting, each party has a private key that is kept secret and a public key that is published and accessible to every other party. The party never publishes or transmits its private key to anyone. One of the most common applications for public-key cryptography is ensuring the confidentiality of a transmission. The sender of a message looks up or is sent the recipient's public key, and uses it to encrypt the message. The recipient uses his private key to decrypt the ciphertext received and to obtain the message. This public key setting allows both secure communication and digital signatures. One of the most widely used public key cryptographic schemes is RSA [11].

LEGAL PARTIES: Alice, Bob

MALICIOUS PARTIES: Charlie

INITIAL CONTEXT: The teacher creates a document d in Alice's workspace (i.e., the personal area used by parties to store their documents and cryptographic artifacts) containing the message 'I love you, Bob'. Then, he generates two distinct pairs of RSA keys, one in Alice's workspace and one in Bob's workspace. Finally, he asks Alice to deliver document d to Bob using public-key cryptography and asks Charlie to try to gain access to the content of the message that Alice is about to send to Bob.

EXPECTED PROTOCOL EXECUTION: Alice asks Bob for a copy of his RSA public key. After obtaining it, Alice uses this key to encrypt the document d and sends the resulting text to Bob, who decrypts it by using his private key.

POSSIBLE PROBLEMS: 1. Bob publishes his private key instead of his public key. 2. Alice uses her public key or her private key, instead of Bob's public key, to encrypt the document to send to Bob. 3. Charlie forges a pair of cryptographic keys impersonating Bob and makes the resulting key available to Alice. Alice uses Charlie's public key to encrypt the document and sends it to Bob. Charlie intercepts the document and decrypts it using his private key.

LEARNING TASK: The primary task is to understand how a public-key communication scheme works and the role played by cryptographic keys in these schemes. Moreover, students may also experience the problem of establishing the real identity of the users participating in a communication scheme and the possible security threats due to the usage of keys forged by malicious users.

B. Using Public-key Cryptography in a Voting Scheme

A possible application for public-key cryptography is the implementation of electronic voting schemes. A group of parties $\mathcal{G}$ may be requested to express a preference among a set of possible choices. The choice should be transmitted electronically to a trusted authority in such a way as to guarantee the confidentiality of the vote.

A simple voting scheme can be implemented by having each party write his preference in an electronic document to be encrypted using the trusted authority public key. The encrypted document is then sent to the trusted authority which, in turn, uses its private key to decrypt it. The scheme can be further improved by using digital signatures to verify the identity of the parties that sent in their preferences.

LEGAL PARTIES: Alice, Bob, Charlie, Dave

MALICIOUS PARTIES: Eve

INITIAL CONTEXT: The teacher creates a pair of RSA keys in Dave's workspace. Then, he reveals to all parties the set of possible choices to cast votes on. Finally, he asks Dave to impersonate the trusted authority and asks Alice, Bob, and Charlie to cast their preferences to Dave, and asks Eve to determine which preference has been expressed by each party.

EXPECTED PROTOCOL EXECUTION: Alice asks Dave for a copy of his public RSA key and uses it to encrypt the document containing her choice. The resulting text is sent to Dave, who decrypts it by using his private key.

POSSIBLE PROBLEMS: Eve writes all the possible choices in different documents in her workspace and encrypts these documents using Dave's public key. When one of the legal parties transmits his preference to Dave, Eve intercepts the encrypted document and compares its content to the content of all of her documents. If a match is found, the preference expressed by the intercepted party is revealed.

LEARNING TASK: The primary task is to demonstrate that when the universe of possible messages to encrypt is small, public-key cryptography may be incapable of providing an adequate level of security. In the discussed case, the problem can easily be solved by the voters inserting into the document containing their preferences, additional text (e.g., some random characters) whose content cannot be predicted by malicious parties.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360

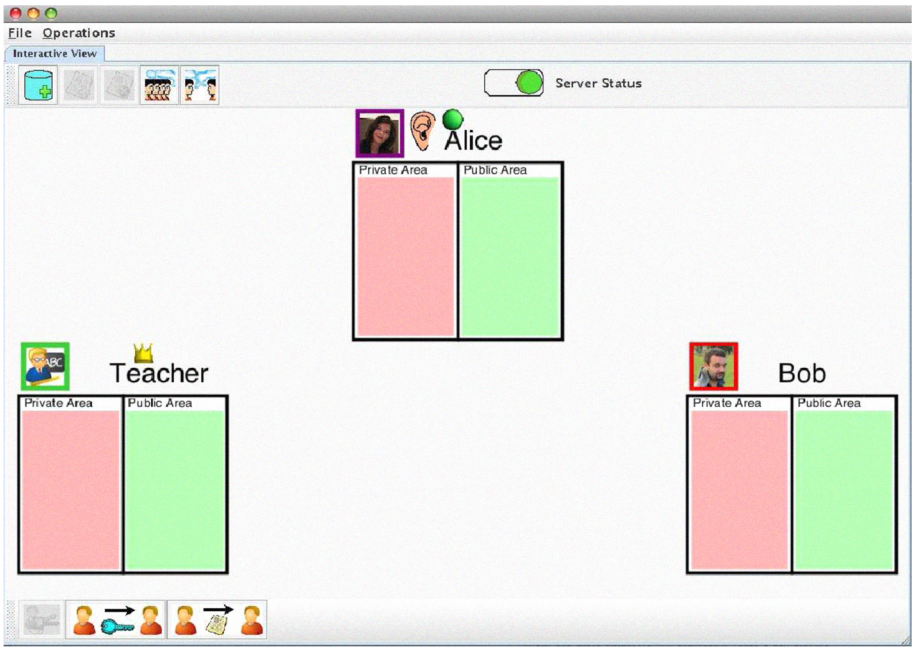

Fig. 1. A screenshot of DISCERN at the beginning of a new session, as seen by Alice. Bob is a legal party. Alice is impersonating a malicious party (recognizable by the ear-shaped icon beside her portrait). The teacher is impersonating himself (recognizable by the crown-shaped icon).

C. *Trusting Public Keys and Certificates*

In order for a public-key secure communication scheme to work, the sender of a message needs the public key of the recipient. Since it is not always possible to assume that this information is available *a priori*, there is the need for a mechanism that would allow the recipient to publish his public credentials, and for the sender to acquire them.

This mechanism can be implemented using Certificate Authorities (CA). These are third parties, trusted by all the participants in a protocol, whose role is to provide digital certificates attesting the correct identities of all the parties involved in a protocol. A digital certificate includes the public key of the party it has been issued to. Another party may verify the integrity and the authenticity of a digital certificate by using the CA public key (which is supposed to be known a priori to all the participants in a protocol). One of the most widely used CA implementation is the one based on X.509 digital certificates [12].

LEGAL PARTIES: Alice, Bob, Charlie

MALICIOUS PARTIES: Dave

INITIAL CONTEXT: First, the teacher creates a document d in Alice's workspace containing the text 'I love you, Bob'. Then, he creates a new certification authority (CA) and uses it to issue two digital certificates: one for Alice and one for Bob. Next, the teacher sends to each participant his digital certificate and the key of the CA. Subsequently, he reveals to Dave that Alice will send Bob a document that Dave has to intercept and read. Lastly, he instructs Alice to deliver document d to Bob using public-key cryptography. *Bob receives istructions to do nothing*.

EXPECTED PROTOCOL EXECUTION: Dave forges a new digital certificate which is apparently entitled to Bob and sends it to Alice. If Alice chooses to test the authenticity of the received certificate by verifying it using the CA public key, she will realize that the certificate has been forged and will stop executing the protocol. Otherwise, Alice will use the key included in the forged Bob's certificate to encrypt message d: the outcoming encrypted message will be sent to Bob. Dave will eavesdrop on the communication channel, acquire a copy of the exchanged message and use the private key he owns to access the message content.

LEARNING TASK: The learning task proves the need for a mechanism to establish the identity of the parties involved in a public-key secure communication scheme. Without this mechanism, it is relatively easy for a malicious party to forge credentials to appear as if they were generated by the recipient of a transmission and to decipher messages encrypted using these credentials.

V. DISCERN

Our system has been implemented in Java as a multi-user distributed application. The simulation of cryptographic protocols takes place in learning sessions where each session is organized in two phases: *setup* and *operational*. In the setup phase, all users (including the teacher) connect to the DISCERN server using a provided application and by choosing a name, a picture, and a role for the party they will impersonate. Users joining a new session are presented with a graphical window visualizing the simulated scenario and all the parties that are connected so far. Each party has a *workspace* that contains all of his artifacts (e.g., cryptographic keys, plain and encrypted documents, digital certificates) accompanied by his name and portrait. Workspaces are split into two areas: a *public*

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360

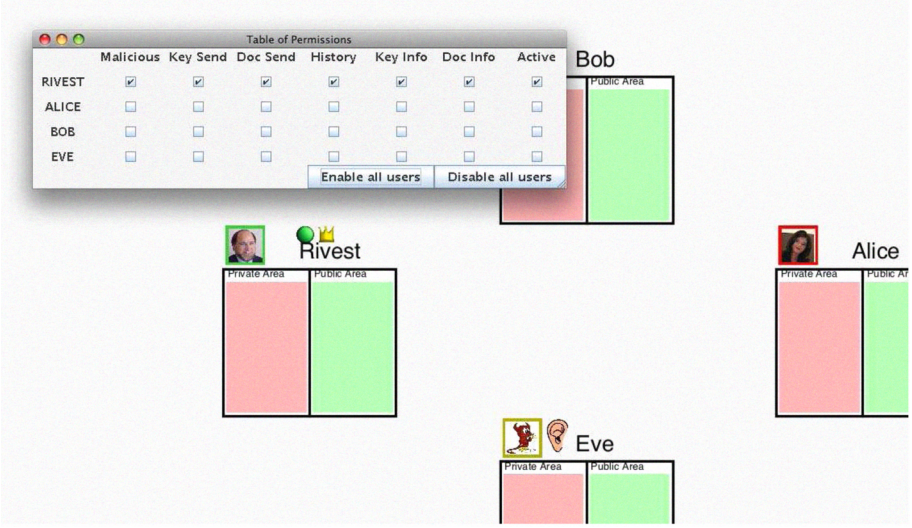

Fig. 2. The control panel the teacher uses to manage the interaction capabilities of all the other parties executing a protocol. In the setup phase the only active party is the teacher himself (i.e., Rivest, in this case).

area, colored green, whose content is visible to all users and a *private area*, colored red, whose content is only visible to its owner and to the teacher. (see Figure 1 for an example).

During this phase, the teacher configures the initial environment for the protocol execution. He can choose which cryptographic primitives will be available during the execution and can manipulate directly the contents of each party's workspace. Once the initial environment is ready, the teacher instructs the parties regarding their goals and advances the session to the following phase.

In the operational phase, the connected parties are free to use their primitives and interact according to the instructions given by the teacher. Users' interactions can be disabled at any time by the teacher, who may temporarily take exclusive control of the session in order to modify the current environment or comment on the current state of execution of the protocol.

Users may join a session by playing one of three different roles:

- **Legal.** It is a party interested in running a cryptographic protocol to share some secret information with other parties under certain security conditions.
- **Malicious.** It is a party interested in disrupting the security of the protocols implemented by legal parties. In addition to the primitives assigned to legal parties, malicious parties also have the possibility of capturing a copy of all the artifacts exchanged by other parties and of forging artifacts that will appear as generated by any of the other parties.
- **Teacher.** This role includes all of the primitives of the malicious and legal parties plus the possibility of ending a session, disabling or re-enabling the interaction capabilities of any of the other parties, or temporarily taking control of the party impersonated by another user. In addition, the teacher can send private or public text messages to the other parties of a session and can choose to add two additional entities to a session: a *public storage area*, to share cryptographic keys, and a *digital certification authority*, to issue and verify digital certificates. A screenshot of the control panel the teacher uses to manage permissions for the other parties is shown in Figure 2.

During a session, the only role that is publicly known is that of the teacher. Moreover, the visualizations of all the actions that may reveal the identity of a malicious party (i.e., eavesdropping on a communication channel) are only visible to that party and to the teacher.

A. The User Interface

The user interface of DISCERN is an evolution of the one used in GRACE and is based on the concepts of *parties* and *artifacts*. A party is an entity that may actively participate in a protocol and is impersonated by a user. As already mentioned in Section V, he is visualized through his portrait and his workspace. The artifacts are the digital information needed for the execution of a cryptographic protocol (e.g., an encrypted document, a cryptographic key, a digital signature) and are represented using the visual metaphors traditionally used for these concepts, as shown in Figure 3. Each artifact may exist in multiple copies and may be found in the public or the private area of one or more workspaces. A party can make his copy of an artifact public by dragging the icon representing it over the green area, and vice-versa. Moreover, each artifact has a set of information describing its properties that can be accessed by right-clicking on it. This information is usually established during the artifact's creation and includes the identity of the party that created that artifact. Malicious users can forge this information and make the artifact appear as if it was generated by any other of the parties connected to the same session. Parties and artifacts have the following set of associated tasks:

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360

Fig. 3. The workspace of Alice. Her private area contains her RSA private key, Bob's RSA public key and an unencrypted document. Her public area contains her RSA public key and an encrypted copy of her document. She is now engaged in forging a new pair of cryptographic keys and, therefore, she is choosing the identity of the party to impersonate.

- **Cryptography related tasks.** Cryptography related artifacts may have cryptographic algorithms associated to them. For instance, a cryptographic key may come with two algorithms: one for encrypting a text and another for decrypting a text. These algorithms may be triggered by combining the icon representing a cryptography-related artifact with another artifact. By doing so, the system will pop-up a list of all the cryptographic algorithms implemented by that artifact which are compatible with the destination artifact. After choosing one of them, the system will run the algorithm and will show the mathematical elaborations performed by that algorithm in a text window on all the clients.
- **Communication related tasks.** Parties involved in a protocol execution can exchange artifacts by offering them or by taking them. In the first case, a user drags an artifact from the workspace of his party to the workspace of another party: a confirmation window will pop-up on the destination client, asking the other user if he accepts that artifact. If he agrees, a copy of the artifact is transferred to the private area of the destination party. A user can also drag one of the public artifacts of a destination party over to his workspace: in this case, the transmission takes place immediately, without any further confirmation. In both cases, all the malicious parties participating in the scenario will automatically acquire a copy of the artifacts being exchanged.

A user can play his part in a cryptographic protocol by using his party to instantiate the proper artifacts and run the cryptography and communication-related tasks needed for the correct execution of the protocol in a timely manner.

B. Implementation Notes

Basically, DISCERN has been designed considering two opposite needs. On the one hand, we had to guarantee that the content of a visualization be identical to all users participating in the same session and that simultaneous users' interactions would be processed in such a way as to guarantee the coherency of the simulation. On the other hand, we were interested in designing a system capable of operating efficiently and without particular delays in user interface responsiveness, even when several users were connected at the same time.

These requirements have been fulfilled by designing an architecture where all the users participating in the same session are connected to a centralized server and maintain, in their client application, a local copy of the session. The role of the server is to coordinate and synchronize the protocol execution. On the client side, all the parties and artifacts existing in a session are modelled as objects, and all the actions they support are implemented as methods. These objects exist at two levels. At an application level, they provide a concrete implementation for the concepts they describe (e.g., the object `Document` holds a text document) and for the actions they support. At a visualization level, they are represented using visual metaphors.

Each user is free to interact with his own local copy of the session, browse the workspace of other parties or read the properties of his artifacts without the need to communicate with the remote server. Whenever one of the users asks for an action that has the effect of modifying the content of the

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360

session (e.g., creating a new document), the corresponding request is sent to the server. Once received, the request is initially put in a queue and then eventually processed if the requesting user has the permission to perform that action. When committed, a request is sent to the clients of all users participating in the session, where it will be run both at an application and at a visualization level.

This approach produced several important advantages. Firstly, it is efficient, because it allows a visualization to be updated by just communicating the details of the action to be executed, usually in a few hundred bytes, and not the payload required to describe graphic context changes. Secondly, it guarantees the coherency of the visualization as actions cannot be executed simultaneously. Thirdly, it simplifies the implementation of the different roles supported in DISCERN, as the server trivially decides which request to forward according to the identity of the requester and to his role.

Notice that this solution does not prevent the possibility for users of issuing interaction requests that will not be executed because of actions performed by other users (e.g., a user asks another user for a copy of an artifact that has been, in the meantime, deleted). In these cases, the server side will be unaware of these problems and will still commit these requests to the connected clients that will silently discard them.

With regards to the cryptographic issue, this is an evolution of the one already used in the GRACE system (see [8]). A standard set of cryptographic primitives with associated visualizations has been defined in DISCERN. This set includes several of the cryptographic algorithms commonly taught in security courses and takes advantages of the implementations available in the Java Cryptographic Extension [13] standard package.

The user interested in providing support for other algorithm or cryptographic protocols has to code them as java classes compliant with the corresponding DISCERN interfaces.

VI. Conclusions and Open Issues

In this paper we presented DISCERN, a system to help students learn how to use cryptographic protocols. DISCERN actively engages students in a simulation of a cryptographic protocol, where each student impersonates one of the parties executing the protocol. Our system proposes an approach to the learning of a cryptographic protocol that is collaborative and competitive. It is collaborative, because students impersonating honest parties have to cooperate and coordinate themselves in order to correctly run a protocol. It is competitive because students impersonating dishonest parties will use their primitives to deceive honest parties and take advantages of their errors in order to threaten the security of the protocol being run. Students may collectively experience some of the consequences of these errors and the way they disrupt the security properties of a protocol, because the cryptographic primitives available with DISCERN and used in a protocol execution are not only simulated but concretely implemented.

There are still some significant issues that need to be investigated in our work. First of all, there is need of a deeper study on the effects that this approach may have on the students' understanding of these topics. In our case, we performed several tests with students participating to a Security on Communication Networks undergraduated course during the fall 2008 semester and at debugging and testing the system. In these tests we organized our students in groups, with each group featuring a malicious party whose role was unknown to ther other parties, and asked each group to recreate one of the two application scenarios presented in Section IV and based on the usage of public-key cryptography. During these tests, we observed a positive effect on the students about their understanding of cryptographic protocols, and we experienced that the competitive factor was the main point driving the attention and the actions of the students. However, we are aware there is the need for a more extensive and thorough investigation on the educational benefits of our approach that should compare it with both the approaches based on the usage of electronic-slides and on the usage of non-collaborative cryptographic protocols learning tools.

Another significant issue arisen during our experimentation concerns with the efforts to be spent for preparing a learning session versus the number of students thay may participate in it. On a side, the teacher may have to spend a not-so-small amount of time to setup complex learning sessions. On the other side, the number of students that may be actively involved in a protocol execution at a time is relatively small. So, the implementation of this approach seems to be problematic when dealing with large classes and requires the development of solutions for scaling on the number of students participating in a session without impacting negatively on their learning experience.

There are several possible directions to be investigated to this end. A possible solution would be to make it possible for a teacher to initiate and manage more than one learning session at a same time. Another interesting direction is to allow many students to join a learning session as viewers, without a party to impersonate initially, but with the possibility for the teacher to assign them a party to play at any time during the execution of a protocol.

References

[1] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography (second edition)*. CRC Press, 2001.

[2] A. Chickering and Z. Gamson, "Seven Principles for Good Practice in Undergraduate Education." *AAHE Bulletin*, vol. 3, p. 6, 1987.

[3] J. J. McConnell, "Active learning and its use in computer science," in *ITiCSE '96: Proceedings of the 1st conference on Integrating technology into computer science education*. New York, NY, USA: ACM, 1996, pp. 52–54.

[4] R. Guthrie and A. Carlin, "Waking the Dead: Using interactive technology to engage passive listeners in the classroom," in *Tenth Americas Conference on Information Systems 2004*, 2004, pp. 534–535.

[5] "Protoviz – a simple protocol visualization," http://www.cs.chalmers.se/~elm/courses/security/.

[6] J. Zaitseva, "TECP - Tutorial Environment for Cryptographic Protocols," Master's thesis, University of Tartu, 2003.

[7] D. Schweitzer, L. Baird, M. Collins, and M. Sherman, "Grasp: A visualization tool for teaching security protocols," in *Proceedings of the 10th Colloquium for Information Systems Security Education, June 2006*, 2006.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360

[8] G. Cattaneo, A. D. Santis, and U. F. Petrillo, "Visualization of cryptographic protocols with grace," *J. Vis. Lang. Comput.*, vol. 19, no. 2, pp. 258–290, 2008.
[9] "Jcryptool," https://jcryptool.sourceforge.net/.
[10] T. Greening and R. Lister, Eds., *Teaching secure data communications using a game representation*, ser. CRPIT, vol. 20. Australian Computer Society, 2003.
[11] R. L. Rivest, A. Shamir, and L. M. Adelman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[12] R. Housley, W. Ford, W. Polk, and D. Solo, "RFC 2459: Internet X.509 public key infrastructure certificate and CRL profile," http://www.ietf.org/rfc/rfc2459.txt, January 1999.
[13] "Java cryptography extension," http://java.sun.com/products/jce/.

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8360
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8360