Journal of Computer Science 4 (7): 503-508, 2008
ISSN 1549-3636
© 2008 Science Publications

Animation of Natural Language Specifications of Authentication Protocols

Mabroka Ali Mayouf and Zarina Shukur
Faculty of Technology and Information Science,
University Kebangsaan Malaysia, Selangor, Malaysia

Abstract: Problem Statement: A few visualization tools have been created for protocol design and analysis. Although these tools provide an environment for designing security protocols, each one has its own protocol definition language (its also called informal specification language). The problem is that the user should understand the language which related to the used tool in order to define an exist protocol or design a new one. For specification, a language needs to be intuitive as well as easily usable and understandable by the security protocol engineer. It must be able to precisely and unambiguously specify the behavior of security protocol. **Approach:** In this study, we propose an approach for protocol specification based on the Natural language definitions of protocol semantics. By using programmatic semantics together with animations, representational flexibility of different protocol demonstration is retained for as long as it is needed. **Result:** This study provides an environment that can be used by protocols designers to develop and investigate different scenarios of security protocols especially authentication protocols. Natural Language Protocol Specifications (NLPS) approach is used to define the protocol. The environment accepts the natural language text of protocol specifications and converts it to animations of protocol behavior. **Conclusions/Recommendations:** NLPS environment can really help protocol designer to consider and investigate the behavior of security protocols. It can also be used for teaching-learning security protocol concepts. In further, we will consider the possibility of analyzing security protocols using our NLPS environment and animation techniques in order to improve the correctness; that is determining whether or not the intended security properties of a protocol do hold.

Key words: Animation, natural language programming, authentication protocol, protocol specifications, storytelling

INTRODUCTION

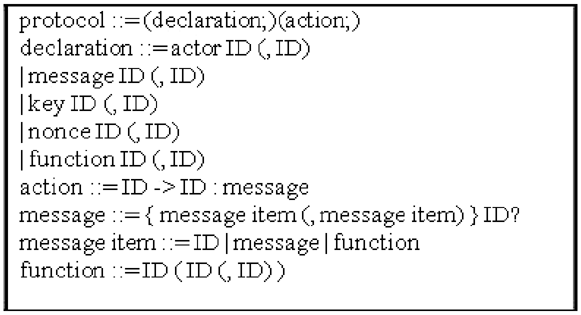
In the field of software engineering, graphical and diagrammatic representations are used to support many activities of the software development process such as specification design and analysis. Interactive visual representation of requirements specifications will help users to create, review and understand formal specifications which are used for protocol analysis. Providing an animated interactive environment for design security protocol will improvise readability and understandability of secure communication between individuals.

An authentication protocol is an exchange of messages having a specific form for authentication of principals using cryptographic algorithms [5]. Specifications that describe and design authentication protocols are usually very complex to read and understand with a lot of different entities of different types being passed back and forth between actors in the system. Requirements specifications of authentication protocols are usually described in an informal standard notation. The standard notation specifies these types of protocols at high level of abstraction indicating the order, the direction of flow and the contents of protocol messages.

However, informal standard notation does not explicitly state the necessary actions to verify messages received by neither the principals nor the meaning of the message contents. Furthermore, many informal standard notations have been created to describe the specifications of protocol and users confused as to which one should be used for describing protocol. Although tools for designing and analyzing security protocols exist[1-3] , the definition of the protocol specifications are still written informally.

Natural language definitions of protocol semantics are intuitive but they are inherent ambiguities.

Corresponding Author: Mabroka A. Mayouf, Faculty of Technology and Information Science, University Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia Tel: +60162320942, +60126623470 Fax: +60390575075



```
protocol ::=(declaration;)(action;)
declaration ::=actor ID (, ID)
|message ID (, ID)
|key ID (, ID)
|nonce ID (, ID)
| function ID (, ID)
action ::=ID -> ID : message
message ::= { message item (, message item) } ID?
message item ::=ID | message | function
function ::=ID (ID (, ID) )
```

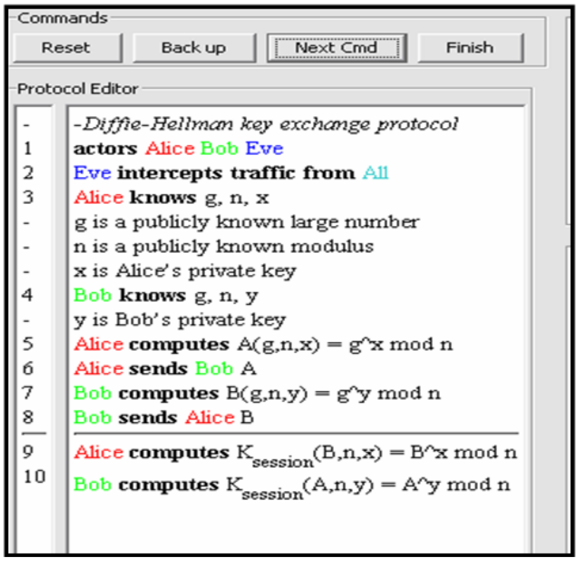
Fig. 1: Extend BNF ProtoViz specification language

While some see the inherent "ambiguity" of natural language as a problem, we see it as an important advantage[4]. By using natural language understanding to construct the mapping between natural language protocol specifications and object-oriented programming language details on a dynamic basis, representational flexibility is retained for as long as it is needed.

From storytelling viewpoint, it is true that every program tells a story. Programming, then, is the art of constructing a story about the objects in the program and what they do in various situations[4]. From the definition of authentication protocol[5], it can also be mentioned that every protocol tells a story. So, it is possible for natural language protocol specifications to be converted into programs and animations in order to demonstrate the behavior of protocol. Animation technique facilitates ''reuse'' more naturally, since the basic components and functions of security protocols and animation can be reused for several different protocol animation scenarios. This is evidenced by the rapid prototyping capability of computer and video games where although the characters and story lines change, the basic animation remains constant[6].

This research describes the possibility of an animation of Natural Language Protocol Specifications (NLPS) of authentication protocol by presenting an environment which help protocol designer to create, review and understand the behavior of security protocols.

Theoretical background: Many animated visualization tools has been developed in computer sciences education. There is a widespread belief among computer educators that visualization technology can improve the effectiveness of learning[7]. Although tools for illustrating computer networking and security protocols concepts exist[8-11], users are not able to add or modify these tools in order to consider different concepts or protocols.



-*Diffie-Hellman key exchange protocol*
actors Alice Bob Eve
Eve **intercepts traffic from** All
Alice **knows** g, n, x
g is a publicly known large number
n is a publicly known modulus
x is Alice's private key
Bob **knows** g, n, y
y is Bob's private key
Alice **computes** A(g,n,x) = g^x mod n
Alice **sends** Bob A
Bob **computes** B(g,n,y) = g^y mod n
Bob **sends** Alice B
Alice **computes** $K_{session}$(B,n,x) = B^x mod n
Bob **computes** $K_{session}$(A,n,y) = A^y mod n

Fig. 2: GRASP commands for Deffie-Hellman protocol

For protocol design, few tools have been developed. One of the protocol visualization tools is ProtoViz which is reported by Elmqvist[1]. This tool allows arbitrary protocols to be described in a step-by-step fashion. The visualization consists of animated message packets moving back and forth between actors. It accepts a protocol description specified using a simple language (Fig. 1) and then transforms its entities and actors into a visual form.

However, this language is informal and user should understand this language in order to define and specify his protocol. GRASP[3] is another tool which employs a similar approach to ProtoViz. It describes a protocol by using a simple protocol specification language that allows an arbitrary number of actors and message passing. The language is also informal and slightly more complex than ProtoViz (Fig. 2).

Most of the tools have its own protocol specifications definition and users are not sure which one is more suitable to be used. In this research, an environment for protocol design is provided which most users who study security protocol concepts can utilize almost immediately.

MATERIALS AND METHODS

The story is a concept used by several research communities in several different ways. In regards to both security protocol specifications and programming storytelling, a story is hereby defined as a series of scenes or events.

Table 1: Protocol actors

Name	Explanation	Visual form
Alice	First participant in all the protocols	
Bob	Second participant in all the protocols	
Carol	Participant in the three- and four party of protocols	
Dave	Participant in the four party of protocols	
Eve	Eavesdropper (someone is listening to your message)	
Mallory	Malicious active attacker (someone is stealing your message and making change to it)	
Trent	Trusted arbitrator (it might be a server, in this research protocols will be called a Key Distribution Center (KDC))	

Table 2: Operations on message

Method	Description
Create message	Create a message box
Generates	Create value
Extracts	Get value from the message box
Concatenates	Put the information into the message box
Encrypt	Lock the message box with suitable key
Decrypt	Unlock the message with the key that it is locked by
Sign	Sign the message with the suitable key
Verify signature	Focus on the signed message
Verify value	Focus on the verified value
Confirm value	Value is equal to the value that has been sent last

A protocol is a series of steps, involving two or more parties, designed to accomplish a task[12]. From this definition, a "series of steps" means that the protocol has a sequence of "scenes" from start to end. Every step "scene" must be executed in turn. "involving two or more parties" means that at least two people "actors" are required to complete the protocol. A person alone does not make a protocol.

If the protocol is organized as a series of scenes, the execution of the protocol proceeds linearly through the scenes and each scene involves at least one of two things: computations by one or more of the actors, or messages sent among the actors, then this organization will be equivalent to a story.

Natural Language Protocol Specifications (NLPS): the first step in designing security protocol is to define the protocol specifications using text form (natural language texts), then translate these texts into informal specifications using the standard notation. A simple protocol, where actor A (denoted to Alice) sends actor B (denoted to Bob) a message consisting of her identifier A and a random number (it is also called nonce) R_A, all encrypted E with B's public key$_{KB}$, would be specified as:

$$A \longrightarrow B: E_{KB}\{A, R_A\}$$

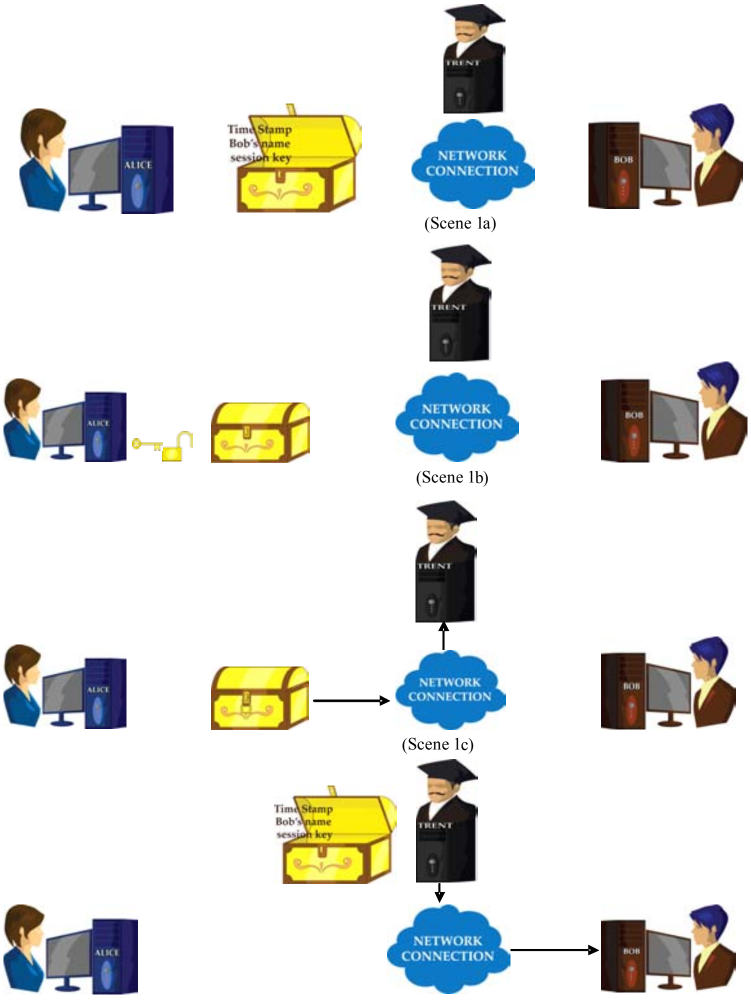
The notation above is informal. It does not explicitly state the actions necessary to verify messages received by the actors, nor the meaning of the message contents. For specification, a language needs to be intuitive as well as easily usable and understandable by the security protocol engineer. It must be able to precisely and unambiguously specify the behavior of security protocol. Natural language definitions of protocol semantics are intuitive, but they are inherent ambiguities. By using programmatic semantics[13] together with animations, representational flexibility of different protocol demonstration is retained for as long as it is needed.

The concepts of writing natural language protocol specifications is based on the strategy used by Schneier[12] . In order to demonstrate protocols, enlisted are the aid of several actors (Table 1). Alice and Bob are the first two. They will perform all general two-person protocols. As a rule, Alice will initiate all protocols and Bob will respond. If the protocol requires a third or fourth person, Carol and Dave will perform those roles. Other actors will play specialized roles as needed.

Rules of writing natural language protocol specifications: According to local environment, the script of protocol specifications must be written as follows:

- Each protocol has its own script
- Each script consists of a series of scenes
- The execution of protocol proceeds linearly through the scenes
- Each scene involves two actors and one message sent
- Message content is explained during the scene
- Operations on messages (Table 2) should also be explained during the scene.

An example interaction: Below is a demonstration of a run through of a brief scenario using Wide-Mouth Frog protocol[14]. This protocol is the simplest symmetric-key cryptography and Trent (a trusted server). Both Alice and Bob share a secret key with Trent. The keys are just used for key distribution and



(Scene 1a)

(Scene 1b)

(Scene 1c)

(Scene 2)

Fig. 3: The Animated scenes of wide-mouth frog protocol

not to encrypt any actual messages between users. To demonstrate this protocol, first, the user chooses actors from actor's gallery before writing protocol script. The actor's gallery consists of images for each actor which has been enlisted in Table 1. Second, the user writes the protocol specifications as a script as below:

- Alice concatenates a timestamp, Bob's name and a random session key and encrypts the whole message with the key she shares with Trent. She sends this to trent, along with her name
- Trent decrypts the message from Alice. Then he concatenates a new timestamp, Alice's name an the random session key. He encrypts the whole message with the key he shares with Bob and then sends this message to Bob

The input two scenes to our environment are parsed using the MontyLingua natural language understanding system[15], the system first performs a surface parse of each input scene (one sentence or more) into VSOO (verb-subject-object-object) form, then semantic recognizer mulls over the VSOO to identify existing object-oriented semantic where each verb is interpreted as a method and nouns as an object.

A simple class is built and passed to the action animator engine to animate the appropriate methods upon the chosen actors from the actor's gallery. For the previous example, the class which is built for the first scene will be as:

```
Class scene1{
SENDER=”Alice”;
RECEIVER=”Trent”;
MSG1[]="Alice's name",
ENMSG;
ENMSG[]="Alice's keyshare"
"timestamp",
"Bob's name",
"session key",
Public void animate (SENDER, RECEIVER,
MSG1)
{
Animate(SENDER, RECEVIER, MSG1)
}
}
```

The second scene class is as:

```
Class scene2{
SENDER="Trent";
RECEIVER="Bob";
MSG1[]=ENMSG;
ENMSG[]="Bob's keyshare"
"Bobtimestamp",
"Alice's name",
"session key",
Public void animate (SENDER, RECEIVER,
MSG1)
{
Animate(SENDER, RECEVIER, MSG1)
}
}
```

The animated scenes progressively demonstrate the idea of the considered protocol is shown in Fig. 3.

RESULTS

NLPS environment: In this study an environment for designing and describing security protocols, especially authentication protocols, is proposed. The introduction of NLPS approach can be used to design protocol specifications. The possibility of designing a new protocol is strongly provided. The environment consists of six components which are scene/script editor, actor's gallery, parser, semantic recognizer, action animator and protocol demonstration storage. The conceptual design of our environment is shown in Fig. 4.

Scene/script editor: The role of this editor is to enable users to write NLPS as texts. Each protocol consists of a series of sequence scenes. Each scene demonstrates at least two actors and one way message sent.

Actor's gallery: User selects actors from actors' gallery. The important actors of protocol demonstration are enlisted in Table 1. As a rule, Alice should initiate all protocols and any other actors may be selected to respond.

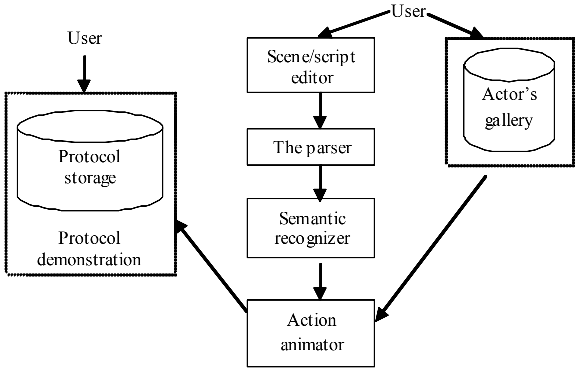


Fig. 4: The conceptual design of NLPS environment

The parser: The MontyLingua natural language understanding system[15] is used to parse each input script into VSOO (verb-subject-object-object) form.

Semantic recognizer: Semantic recognizer mulls over the VSOO to identify existing object-oriented semantic where each verb is interpreted as a method and noun as an object. A simple class is built and passed to action animator component.

Action animator: The passed objects/methods class is linked to the action animator to animate the protocol specifications (actors, operations on message, message passing and messages contents).

Protocol demonstration storage: In order to view the behavior of the protocol, storage is required so that the protocol can be demonstrated anytime.

DISCUSSION

The idea of NLPS approach is based on the Programmatic Semantics of natural language and storytelling approach. Programmatic Semantics is a mapping between natural linguistic structures and basic programming language structures, by taking the position that programming is storytelling[13].

The presented concepts of writing natural language protocol specifications is based on the strategy used by Schneier[12] . Therefore, protocol is hereby considered as a series of scenes and each scene consists of one or more sentences which involve two or more actors and one message passing. Each sentence is parsed into VSOO. Each verb is considered as a method which will be animated together with the passed message. By using programmatic semantics together with animations, the representational flexibility of different protocol demonstration is retained as long as it is needed. An animation of NLPS certainly has the capability of improving the readability and understandability of protocol behavior and security protocol concepts.

CONCLUSION

Our environment provides the possibility of converting NLPS to classes where each verb is interpreted as a method and noun as an object. Then, action animator will animate the proper objects and methods to demonstrate the behavior of the considered protocol. We believe that such environment can really help protocol designer to consider the behavior of security protocol. The possibility of analyzing security protocols using our NLPS environment will be considered as a future work.

ACKNOWLEDGMENT

We acknowledge MontyLingua and it's author Hugo Liu (http://www.web.media.mit.edu/~hugo/montylingua).

REFERENCES

1. Elmqvist, N., 2004. ProtoViz: A simple security protocol visualization. http://www.cs.chalmers.se/~elm/courses/security/report.pdf.
2. Saul, E., 2001. Facilitating the Modeling and Automated Analysis of Cryptographic Protocols. Master thesis, University of Capetown, SA. http://www.cs.uct.ac.za/Reasearch/DNA/SPEAR2.
3. Schweitzer, D. and W. Brown, 2007. Interactive visualization for the active learning classroom. ACM SIGCSE Bull., 39: 208-212. http://doi.acm.org/10.1145/1227504.1227384.
4. Liu, H. and H. Lieberman, 2005. Metafor: Visualizing stories as code. Proceeding of the 10th International Conference on Intelligent User Interface, Jan. 10-13, ACM Press, New York, USA., pp: 305-307. http://doi.acm.org/10.1145/1040830.1040908.
5. Syverson, P.F. and I. Cervesato, 2001. The logic of authentication protocols. Lecture Notes Comput. Sci., 2171: 63-137. http://www.springerlink.com/content/5hj434d3m0pc7eln/.
6. Chen, D.J., W.C. Chen and K.M. Kavi, 2002. Visual requirement representation. J. Syst. Software, 61: 129-143. DOI: 10.1016/S0164-1212(01)00108-X.
7. Naps, T.L., G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A.K. Helsinki, L.M. Helsinki, M. McNally, S. Rodger and J.Á. Velázquez-Iturbide, 2003. Exploring the role of visualization and engagement in computer science education. ACM SIGCSE Bull., 35: 131-152. http://doi.acm.org/10.1145/782941.782998.
8. Baxley, T., J. Xu, H. Yu, J. Zhang, X. Yuan and J. Brickhouse, 2006. LAN attacker: A visual education tool. Proceeding of the 3rd Annual Conference on Information Security Curriculum Development, Sep. 22-23, Kennesaw, Georgia, pp: 118-123. http://doi.acm.org/10.1145/1231047.1231072.
9. Holliday, M.A., 2003 Animation of computer networking concepts. J. Educ. Resource. Comput., 3. http://doi.acm.org/10.1145/982753.982755.
10. Yuan, X., Y. Qadah, J. Xu, H. Yu, R. Archer and B. Chu, 2007. An animated learning tool for Kerberos authentication architecture. J. Comput. Sci. Coll., 22: 147-155. http://portal.acm.org/citation.cfm?id=1231091.1231116.
11. Yuan, X., P. Vega, J. Xu, H. Yu and Y. Li, 2007. Using packet sniffer simulator in the class: Experience and evaluation. Proceedings of the 45th Annual Southeast Regional Conference, Mar. 23-24, Winston-Salem, North Carolina, pp: 116-121. http://doi.acm.org/10.1145/1233341.1233363.
12. Schneier, B., 1996. Applied Cryptography. 2nd Edn., Published by Wiley and Sons, Inc., USA., ISBN: 0-471-12845-7, pp: 21-23.
13. Liu, H. and H. Lieberman, 2005. Programmatic semantics for natural language interfaces. Proceedings of the ACM Conference on Human Factors in Computing Systems, Apr. 2-7, ACM Press, Portland, OR., USA., pp: 1597-1600. http://doi.acm.org/10.1145/1056808.1056975.
14. Burrows, M., M. Abadi and R. Needham, 1990. A logic of authentication. ACM Trans. Comput. Syst., 8: 18-36. http://doi.acm.org/10.1145/77648.77649.
15. Liu, H., 2004. MontyLingua: An end-to-end natural language processor with common sense. web.media.mit.edu/~hugo/montylingua.