Proceedings of the 12th Colloquium for Information Systems Security Education
University of Texas, Dallas
Dallas, TX June 2 - 4, 2008

A Digital Lego Set and Exercises for Teaching Security Protocols

Weichao Wang, Aidong Lu, Li Yu, Zhiwei Li
University of North Carolina at Charlotte

Abstract – To bridge the gap between the instruction of security primitives and protocols, we have designed and developed a digital Lego system and supporting course materials. Our digital Lego pieces are designed to use shapes to provide a generic representation of security protocols. With the automatic Lego piece generation and fitting method, we have developed a protocol demonstration and experiment environment that allows students to practice with these abstract concepts. The developed exercises will expose the relationship among security primitives and properties, and train students' capabilities to design secure protocols under different requirements. Our approach applies the pedagogical methods learned from toy construction sets by treating security atomics as Lego pieces and protocols as construction results.

Index terms –Digital Lego System, Exercises, Security Protocol Instruction, Education Visualization

I. INTRODUCTION

With the importance of information security being well recognized, more and more universities have established their security curricula [1]. These security curricula usually consist of an introductory level course and a group of advanced courses. The introductory course generally focuses on the **security atomic units** and standard operations such as encryption, authentication, integrity protection, and freshness preservation, which can be viewed as building blocks of information assurance. The advanced courses focus on **security protocols** in specific fields such as operating systems, networks, or databases, and the prevention and detection of attacks on them.

Although this curriculum structure has considered the knowledge development patterns of students, there still exists a gap between teaching security atomic units and teaching security protocols. This gap may impact students' capabilities to design and assess security protocols that can satisfy various combinations of requirements in different applications. It has been shown that a group of atomic units may finally compose vulnerable protocols if they are inappropriately organized. At the same time, students in the advanced courses may spend a significant amount of time on understanding how complex systems (e.g. TCP/IP) work, which may distract them from investigating the relationship among atomic units and protocols. Therefore, a new approach must be designed and developed to bridge this teaching gap. We believe that new instructional tools should be developed to cultivate students' capability to select suitable atomic units and organize them appropriately.

The objective of this project is to develop an innovative digital construction set that integrates the achievements in security education and visualization, and based on this, design instructional demonstrations and hands-on experiments to assist students to bridge the security atomic units and protocols. The approach applies the pedagogical methods that have been learned from the successful education of children and adults using electronic blocks or construction sets [2]. It treats the atomic units as Lego pieces and the protocols as construction results. We have developed three groups of hands-on exercises to help students understand the relationship among atomic units and protocols. We also design exercises with multiple difficulty levels to cultivate their capabilities to manipulate atomic units and design security protocols.

The contributions of the project are as follows. While the decomposition and construction exercises using toy building sets have been shown to be successful in children's education, this project provides an environment to evaluate the effectiveness of this pedagogical method in adult education in information assurance. We carefully integrate security and visualization techniques to improve the outcomes of information assurance education. Our results will also enable instructors to develop and improve new materials for security courses in the future.

We have developed a prototype system that includes atomic unit and protocol representation, Lego piece design, protocol construction, user interactions, and corresponding demonstrations and exercises. The remainder of the paper is organized as follows. Section II discusses related work. In Section III we introduce the overall architecture of the Lego system and the details of each component. Section IV presents the design of demonstrations and hands-on experiments. Section V discusses future extensions. Finally, Section VI concludes the paper.

Contacts: weichaowang, aidong.lu, lyu8, zli19@uncc.edu

II. Related Work

Construction sets have an important place in the history of education. A building set for castles and walled towns appeared as early as in 1800 [3]. In America, building blocks have been recommended to parents since 1826 [4]. The educational role of construction sets has been enhanced through the integration of computational media. Building blocks with sensors and fiber optic output can be used to construct a speech-enabled alphabet set [5] or 3D structures that can communicate to a computer [6].

Construction sets have been widely used in undergraduate robotics education. For example, LEGO bricks [7] can be used as controllers for large LEGO sets. The sets provide a wide space for students to make hypotheses about how things work and validate their assumptions [8]. Similar digital construction sets have been used in artificial intelligence, programming, and general engineering courses [9, 10, 11]. Inspired by the success in robotics education, digital construction sets have been applied to the design of space habitat and vehicle [12] and computer systems[13].

This proposed research is inspired by the fact that various security protocols are constructed by a limited number of primitives. For example, Millen and Shmatikov [14] have summarized ten reduction rules to decompose security protocols into simple units. Cremers [15] investigated how to decompose a complicated protocol into subprotocols. A graphic environment for security protocol analysis is proposed by Saul and Hutchison [16].

Research efforts have been made to assist the teaching of abstract security protocols. For example, Saul and Hutchison [17] have proposed a graphical tree-based specification environment to teach GNY based security protocol analysis. Hamey [18] developed a highly visual and interactive game environment for teaching secure data communication protocols. Bergstrom et al [19] developed a visual learning environment to teach network security courses. A visualization tool for teaching security protocols is proposed by Schweitzer et al [20].

III. Infrastructure of Digital Lego System

We have developed a new education tool for teaching security atomics and protocols using digital Legos. We visualize security protocols with Lego pieces to enhance important concepts in security curriculum and provide an interactive environment for both lectures and exercises. Our Lego construction and demonstration system integrates advanced techniques from several research areas such as security protocol verification and interactive visualization. We first briefly describe the working procedure of the system and provide more details of each component in the following subsections.

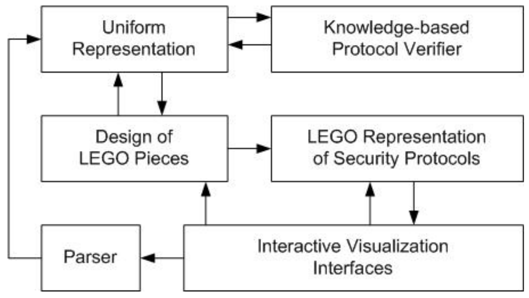

Uniform Representation
Knowledge-based Protocol Verifier
Design of LEGO Pieces
LEGO Representation of Security Protocols
Parser
Interactive Visualization Interfaces

Figure 1. Architecture of our digital Lego system.

Figure 1 illustrates the overall infrastructure of our Lego system. The system provides two channels for users to input and construct security protocols: plain text and an interactive visualization interface. In the first method, a user can type text to describe messages exchanged among protocol participants. These messages will be parsed into a uniform representation format and stored in the system. In the second method, a user can directly construct security protocols through dragging-and-dropping a group of pre-defined Lego pieces. The input security protocol will be illustrated as Lego construction results. The protocol will be passed to knowledge based verification component, which will analyze it and return results to the visualization interface. If the analysis shows that the protocol is not safe, a possible attack will be generated and corresponding Lego structures will be constructed automatically to demonstrate the compromise.

A. Uniform Protocol Representation

We propose a two-level representation for security protocols to assist the design of Lego pieces and future protocol manipulation. At the high level, we use parametric strand space model [21, 22, 23] to describe a protocol. To define briefly, a strand represents the operations of a principal and all messages received by and sent from it. In this way, a security protocol can be represented as strands of principals matching together. Similarly, an attack can be represented as strands of malicious entities inserted between those of legitimate parties. A legible security protocol must satisfy the following two rules: (1) every message must have a sender and a receiver; (2) the sender can generate all components of a message based on its knowledge. We adopt a variation of Athena [22] to analyze security protocols.

We use a free term algebra illustrated in Figure 2 to define messages in a protocol. We define variables as a special kind of atomic terms. To assist substitution operations in protocol verification, we further divide variables into three different types and define them as follows. A *term variable* represents a variable that can be bound to any other terms or an empty term. A *principal variable* can be bound to either a particular principal name or another principal variable. A *key variable* can be bound to a public key, a private key, a symmetric key, or another key variable.

Atomic	::=*Principal Name* | *Nonce*| *Key* | *Variable*
Key	::=*Public Key* | *Private Key* | *Symmetric Key*
Variable	::=*Term Var.* | *Principal Var.* | *Key Var.*
Encrypted	::=$\{Term_1\}_{Term_2}$
Concatenated	::=$Term_1 \wedge Term_2$
Term	::=*Atomic* | *Encrypted* | *Concatenated*

Figure 2. Syntax of term representation.

B. Knowledge Based Protocol Verification and Attack Construction

In this system, we propose a generic framework to model knowledge of legitimate parties and attackers and describe the procedures of knowledge learning and reasoning. To define briefly, a knowledge model consists of two essential components: a *knowledge base* that contains the minimum set of information units known to the entity, and a set of *rules* to learn new knowledge or apply existing information.

Deriving the minimum set of information units to form the knowledge base is crucial to the termination and efficiency of protocol verification since a poorly represented knowledge set may lead to state explosion during the analysis procedure. The generation and usage of this knowledge base depend on an inference system, which consists of two sets of rules. The reduction rules try to decompose received or eavesdropped messages through a group of operations such as splitting or decryption to learn new knowledge. The deduction rules, on the contrary, try to synthesize messages based on the information units in the knowledge base. Although the internal representation of knowledge bases of a legitimate party and an attacker is similar, they have different rule sets in their inference systems. For example, a legitimate party will not use an identity to replace a random number with the same length. An attacker, on the contrary, will conduct such an operation to form a type-flaw attack.

Using strand space models, Perrig and Song [24] have generated a group of well-formed formula to represent the most widely used security properties such as authentication and confidentiality. In this way, the procedure of protocol verification works as follows. We will start from the strand of an innocent participant and formulate it into an initial state. Using the inference systems of legitimate parties and attackers, we try to construct all possible complete message exchanging procedures that follow the format of the protocol. Finally, we will identify those complete procedures that do not contain corresponding strands of other legitimate participants and generate attack procedures.

C. Design of Lego Set

To effectively visualize security protocols, we design a set of digital Lego pieces that can transform abstract protocol information into meaningful visual forms. Our design is to use the boundary shapes of Lego pieces to represent all the protocol information. Simulating the key features of real Lego sets, two digital Lego pieces can be put together only when their adjacent boundaries share the same shape. We design Lego pieces to represent common sending and receiving activities and use multiple Lego pieces to visualize a protocol. We believe that this design matches the spirit of Legos closely, which will increase the interests of students and attract their focus to important security concepts represented by Lego shapes.

We have developed an automatic Lego generation approach that is generic for visualizing various security protocols. Our approach is to design basic shapes to cover all the atomics in the uniform representation of protocols, and use them to synthesize Lego pieces by connecting basic shapes. As shown in Figure 3, every Lego piece is generated through composing multiple blocks. We use two green blocks and one adjustable blue block as the basic shape of a Lego piece. The shapes of the two green blocks match each other to ensure the vertical connection of a strand. They always point downward, since we assume that protocols are executed from top to bottom. The two orange blocks on the left and right sides are used to represent message contents. Their size is automatically adjusted according to the length of messages. A Lego piece can represent the sending of a message with convex shapes or receiving with concave shapes. This design allows us to generate all the Lego pieces automatically and compose various shapes freely.

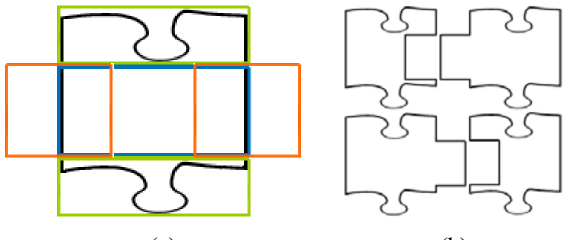


Figure 3. Design of protocol Legos. (a) The composition of a Lego piece; (b) The top row shows an example of Lego pieces representing message delivery when sender is on the right side; the bottom row shows an example of message delivery when sender is on the left side.

To represent message contents, we align meaningful textures in 2D message grids to generate multi-level Lego boundaries. We design a general construction method based on the message grammar, so that we can visualize all kinds of message contents. From the message grammar shown in Figure 2, we can see that a message can include a list of atomics connected by manipulation operators such as encryption. We can view concatenation as the connection of two or more atomics at the same level, while encryption as the coverage of atomics at a deeper level. For example, in a message $\{A, \{A, B\}_{KAB}, B\}$, the top level contains "*A*", "K_{AB}", and "*B*", and the second level contains "*A*" and "*B*" that are covered by encryption. In this way, we can transform any message into a 2D grid tree structure.

Each element in the 2D grid can be uniquely represented by two properties: its category and related principal. We have summarized several categories based on the grammar, such as principal, nonce, public keys, symmetric keys, and data. We assign a different shape to each of these categories. Similarly, we assign a different shape to each principal name. This two-feature representation of an element is more efficient than generating a different shape for every principal-category combination. As shown in Figures 4, 5, 6, each atomic unit is represented by its category texture and related principal on the top. We choose intuitive shapes for category textures, shown in Table 1. For example, we use character "*K*" for keys. We always place higher levels outside, so that only when the outside boundary matches, inside boundaries will be compared. We also add background line patterns to connect multi-level boundaries, so that all the textures are connected as continuous boundaries to represent various message contents.

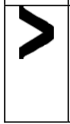
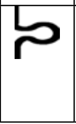
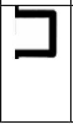
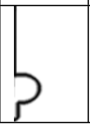
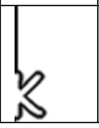
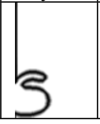
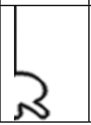
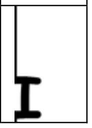
A	B	S	Prin-cipal	Public Key	Sym-metric key	Non-ce	Non-ce 2
							

Table 1. Sample Lego shapes

D. <u>Design of Interactions</u>

We have developed several interaction tools in our digital Lego system for exploring and editing security protocols. These tools will be integrated with the hands-on exercises described in Section IV.

Since message content is the essential information in a security protocol, our interaction tools concentrate on visualizing and editing message contents of Lego pieces with different levels of detail. As shown in the syntax representation of Figure 2, the message content of each sent and received piece can be structured as a tree. Therefore, the depth of the trees is an intuitive way to control message details in the protocol visualization. As shown in Figure 4, starting from depth 0, we only reveal the principal as a sender or receiver without showing any message contents. At a certain depth, we will visualize all the contents at and above the current depth. This will show complete message contents at the highest depth level. Since our Lego pieces are synthesized automatically during visualization and interaction, we can use the depth of message trees to control displayed information.

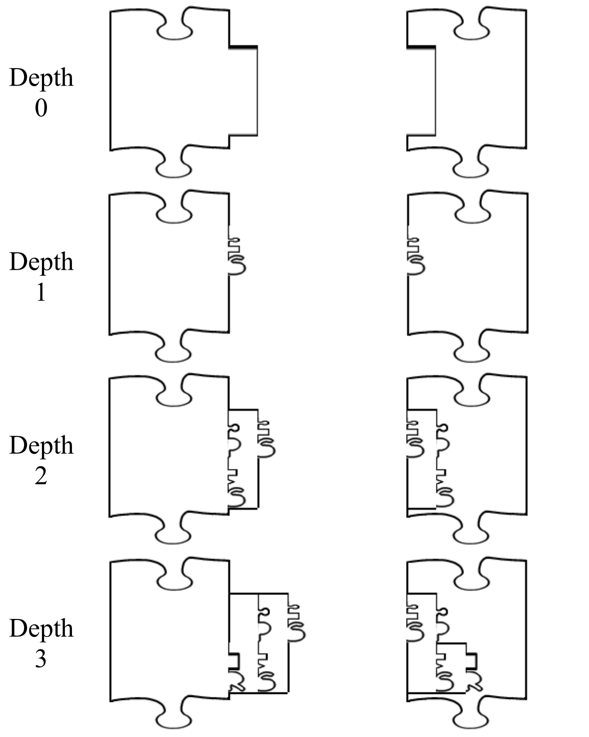


Figure 4. A Lego piece with different levels of detail for message $\{A, \{N_b\}_{KAS}\}_{KBS}$

We provide the following exploration tools for users to visualize protocol contents.

- Overall message details: This tool is designed to control the details of messages in the overall protocol visualization. This tool can help students understand a complex protocol gradually with additional information.
- Group message details: We allow users to adjust message details for principal groups in a protocol. Users can select to adjust the details of senders or receivers, or the details of a specific principal.
- Individual message details: We also allow users to adjust individual message details of sender or receiver Lego pieces.
- Zoom in/out: This tool allows users to enlarge selected pieces or shrink them.

Our editing tools provide the following functionalities:

- Pick: A user can use mouse to point to a Lego piece and click the left button to select it.
- Highlight: A user can highlight an information unit after selection.
- Move: Users can move a Lego piece freely by moving the mouse with the left button pressed.
- New: Users can add a new Lego piece.
- Delete: Users can delete a selected Lego piece.
- Message content modification: The message contents can be modified by editing the text and the system will automatically update the Lego piece.

IV. Hands-on Exercises Using LEGO Sets

We use our digital Lego set to improve security curriculum by developing three groups of hands-on exercises. These exercises are designed with increasing difficulty levels, so that we can integrate them into our lectures and homework according to the required course materials. We design the first exercise group to help students get familiar with security protocols and attacks, the second exercise group to expose relationships among security primitives and properties, and the third exercise group to cultivate capabilities of students to design secure protocols. The following of this Section describes the details of each group and provides examples to illustrate their usages.

A. Automatic Demonstration of Protocols and Attacks

The first step of security protocol instruction is to help students understand the procedures of message exchanges and enforcement of security properties with the execution of a protocol. To achieve this goal, we design our first group of exercises using the proposed Lego system to generate effective demonstrations of a protocol or an attack.

Our system can be used to visualize both pre-loaded and user-defined protocols. The operations of each participant will be laid out as a vertical strand, with each sent or received message as a Lego piece. The knowledge base of each participant will be displayed next to its identity to illustrate the increases as a protocol proceeds. Both students and instructors can use our overall, group, and individual message content controlling tools to adjust the displayed details easily.

While the demonstration of a preloaded protocol or attack can be directly generated, the system will first verify a user-defined protocol before it is accepted. The verification has two purposes. First, the system will make sure that every legitimate party has enough knowledge to generate the messages that it sends out. In other words, we need to verify that the protocol is legible. For example, node *A* will not be able to sign a packet with node *B*'s private key. To enforce this property, the system will start from the initial knowledge base of each participant and verify the changes of its knowledge and the contents of sent messages. The system will ask the user to improve her/his design if a conflict is detected. Second, the system will try to identify vulnerabilities in the protocol, construct possible attacks, and demonstrate them with our Lego set. This task can be accomplished by the protocol verification component following the procedure described in Section III.B. In these demonstrations, the knowledge units for constructing fake messages will be highlighted to help students understand the capabilities of attackers.

To help other instructors and promote the adoption of the Lego system, we have preloaded the system with a group of security protocols and attacks that are widely used in information assurance courses. The following table provides a list of these preloaded contents.

Preloaded Security Protocols
SKID2
SKID3
Wide-Mouth Frog
Yahalom
Needham-Schroeder
Otway-Rees
Woo-Lam
Denning-Sacco
Preloaded Attacks
Man-in-the-middle attack using SKID3
Resend attack using Needham-Schroeder
Type flaw attack using Woo-Lam
Impersonation attack using Denning-Sacco

Table 2. Preloaded protocols and attacks in Lego system.

B. Identifying Critical Information Pieces

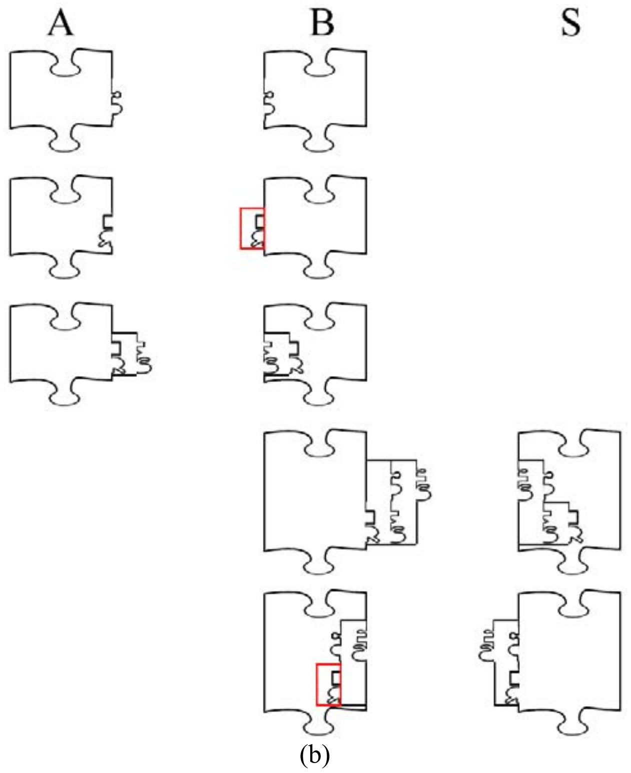
The second group of exercises focuses on training students' capability to identify security primitives in a complex protocol. Although there are hundreds of security protocols that are adopted by different systems, the number of schemes to enforce individual security properties is usually limited. For example, the most frequently used methods to guarantee freshness of information are sequence numbers, timestamps, and freshly generated nonces. Based on this observation, we have identified a group of security properties and enumerated the methods to enforce each of them.

Using this information, we have designed several exercises for each preloaded protocol. In this exercise, students are required to highlight the distributed information units that are essential to the enforcement of a specific security property. After they finish their selection, our system will compare the highlighted contents with previously recorded answers. If the user input is insufficient to enforce this property or more than enough, the system will display the correct answer. Currently, the questions and answers in this exercise are manually designed. We will develop a method to automatically generate questions and answers in the future.

Figure 5 illustrates an example in which the user is asked to identify the contents that help node B authenticate the identity of node A in the Woo-Lam protocol.

$$A->B: A$$
$$B->A: N_b$$
$$A->B: \{N_b\}_{KAS}$$
$$B->S: \{A, \{N_b\}_{KAS}\}_{KBS}$$
$$S->B: \{A, N_b\}_{KBS}$$

(a)



(b)

Figure 5. (a) The Woo-Lam authentication protocol. (b) The highlighted contents illustrate the information that helps node B authenticate the identity of node A.

C. Protocol and Attack Construction with Increasing Difficulties

The ultimate goal of our third group of exercises is to enable students to design security protocols that satisfy requirements of different systems. To achieve this goal, we divide the exercises into three subgroups with increasing difficulty levels. Below we describe each of them in detail.

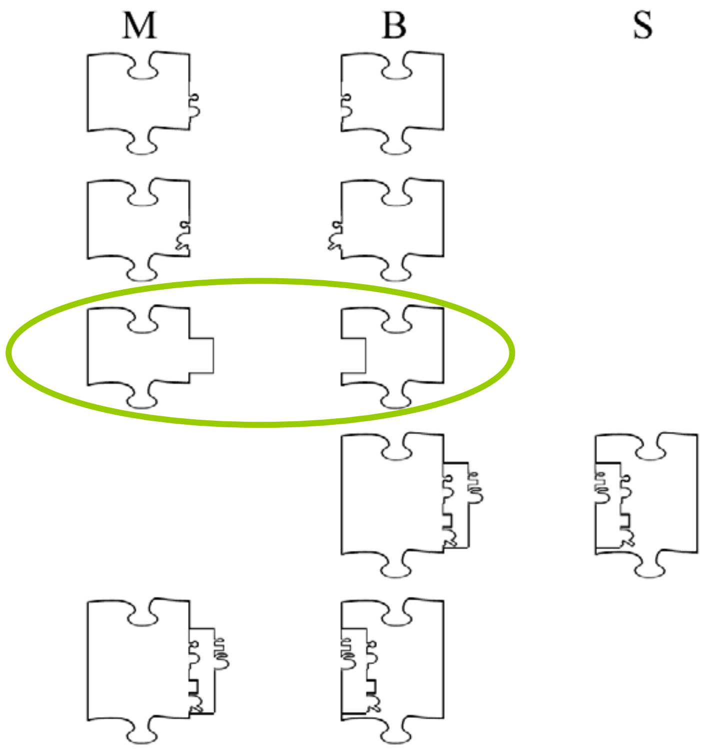
In the first subgroup of exercises, we will remove the information units that are essential for the enforcement of a specific security property and ask the user to fill the blank. These exercises can be viewed as a natural extension of the previous group of practices and they can use the same manually designed questions. Since there may be several different methods to fill the blanks and enforce the property, the user input will be provided to the protocol verification component. If the analysis shows that the protocol is not safe or the property is not enforced, the system will display the result and ask the user to improve her/his design. This procedure will continue until a correct answer is provided by the user or the user chooses to see the system's answer.

The second group of exercises will train students to form messages based on available knowledge. We are especially interested in the formation of fake messages to conduct attacks. In this exercise, the screen will be divided into two parts. On the left side we illustrate the original protocol. On the right side, we will insert a part of the attacker's strand into the protocol execution procedure and leave several messages as blank. The user is asked to fill the lost contents of these messages based on the attacker's knowledge base. After a user fills the blanks, the attack will be analyzed by the system. If the attack fails to compromise the protocol, the system will provide the correct answer. Figure 6 illustrates an example in which the user is asked to form a type-flaw attack on Woo-Lam protocol.

The last group of exercises will train students' capabilities to design security protocols under specific requirements. We will define the initial knowledge bases for legitimate participants and ask users to design a protocol that satisfies one or several security properties. The design result will be provided to the verification component for analysis. Since the initial knowledge base has been identified, a user-defined protocol will be labeled as not legible if it uses some information that is not available to a legitimate party. The verification procedure will also identify vulnerabilities of the designed protocol and construct possible attacks. The user can improve her/his design based on these feedbacks and submit the updated protocol to the verification component for another round of evaluation.

$$M->B:A$$
$$B->M:N_b$$
$$M->B:N_b$$
$$B->S:\{A,N_b\}_{KBS}$$
$$packet\ blocked\ by\ M$$
$$M->B:\{A,N_b\}_{KBS}$$

(a)



(b)

Figure 6. (a) The attacker conducts a type-flaw attack on Woo-Lam authentication protocol. (b) The user is asked to input the fake message sent by the attacker (circled by green lines).

V. Future Work

In the future, we propose to improve the Lego system through strengthening its protocol representation and attack analysis capabilities, providing more user-friendly interfaces, and conducting systematic evaluation. From the technical point of view, we will improve the representation capabilities of the knowledge model by integrating one-way functions into the model. We will also investigate the verification of protocols with non-atomic keys. We will integrate more kinds of security protocols and attacks into the system to assist other instructors and promote its wide adoption.

We will improve the visualization and interaction interfaces of the system so that students can better understand the course materials. For example, previous research [25] has shown that some people will better accept knowledge represented using 2D shapes, while others better using 3D shapes. Therefore, we propose to develop 3D Lego pieces for security protocol construction and verification. We will also develop animation functions that can help to keep students' attention and demonstrate procedures of message exchanging effectively.

With the help from the staff of the Teaching Excellence Center, we plan to conduct an evaluation to assess the effectiveness of the proposed Lego system in helping students understand information assurance knowledge. One of the authors will teach an introductory level security course every year and the course usually attracts 30 to 40 junior level students in computer science and information assurance majors. The evaluation will include a suite of hands-on exercises, team projects, questionnaire, and user interview. The teaching experiences and student feedbacks will provide first hand information for us to improve the approach.

VI. Conclusion

To improve information assurance education, we have developed a digital Lego system for demonstrating and practicing important security concepts. We have carefully designed our digital Lego sets to provide a generic representation of security protocols. Our approach applies the pedagogical methods learned from toy construction sets by treating security atomics as Lego pieces and protocols as construction results. With our digital Lego sets, we have developed a prototype system and supporting instructional materials. Three groups of hands-on exercises have been designed to strengthen different aspects of security education. The system can automatically generate demonstrations and let students perform hands-on experiments. This approach, if widely adopted, will assist instructors in information assurance courses and help students better understand the abstract and challenging materials.

VII. Acknowledgement

This project is supported by National Science Foundation under award numbers 0754592 and 0633150.

VIII. References

[1] V. Pothamsetty. Where security education is lacking. In Proc. of InfoSecCD, pages 54–58, 2005.

[2] P. Wyeth and H. Purchase. Using developmental theories to inform the design of technology for children. In

Proc. of Conf. on Interaction design and children, pages 93–100, 2003.

[3] B. Harley. Constructional Toys. Shire Publications, UK, 1990.

[4] G. Cross. Kids' Stuff. Harvard University Press, 1997.

[5] M. Eisenberg, A. Eisenberg, M. Gross, K. Kaowthumrong, N. Lee, and W. Lovett. Computationally-enhanced construction kits for children: Prototype and principles. In Proc. of Int. Conf. of Learning Sciences, pages 79–85, 2002.

[6] Y. Kitamura and et al. Real-time 3d interaction with activecube. In Proc. of CHI, pages 355–356, 2001.

[7] M. Resnick and et al. Programmable bricks: Toys to think with. IBM Systems Journal, 35(3):443–452, 1996.

[8] J. Weinberg and X. Yu. Robotics in education: Low-cost platforms for teaching integrated systems. IEEE Robotics and Automation, 10(2):4–6, 2003.

[9] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick. Scratch: A sneak preview. In Int. Conf. on Creating, Connecting, and Collaborating through Computing, pages 104–109, 2004.

[10] F. Martin, B. Mikhak, M. Resnick, B. Silverman, and R. Berg. To mindstorms and beyond: evolution of a construction kit for magical machines. In Robots for kids: exploring new technologies for learning, pages 9–33. Morgan Kaufmann Publishers Inc., 2000.

[11] J. Weinberg, G. Engel, K. Gu, C. Karacal, S. Smith, W. White, and X. Yu. A multidisciplinary model for using robotics in engineering education. In Proc. Of The American Society for Engineering Education Annual Conference, 2001.

[12] A. Howe. The ultimate construction toy: Applying kit-of parts theory to habitat and vehicle design. In Proc. of Aerospace Architecture Symposium, 2002.

[13] C. Coulston1 and R. Ford. Teaching functional decomposition for the design of electrical and computer systems. In Proc. of IEEE Frontiers in Education Annual Conference, 2004.

[14] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In Proc. of ACM CCS, pages 166–175, 2001.

[15] C. Cremers. Compositionality of security protocols: A research agenda. Electronic Notes in Theoretical Computer Science, 142(3):99–110, 2006.

[16] E. Saul and A. Hutchison. A graphical environment for the facilitation of logic-based security protocol analysis. South African Computer, (21):26–30, 1998.

[17] E. Saul and A. Hutchison. An Environment to Facilitate the Teaching of GNY-Based Security Protocol Analysis Techniques. In Proceedings Second World Conference in Information Security Education, 2001.

[18] L. Hamey. Teaching secure communication protocols using a game representation, in Proceedings of the fifth Australasian conference on computing education, pages 187-196, 2003.

[19] L. Bergstrom, K. Grahn, K. Karlstrom, G. Pulkkis, P. Astrom. Teaching Network Security in a Virtual Learning Environment, Journal of Information Technology Education, vol 3, pages 189-217, 2004.

[20] D. Schweitzer, L. Baird, M. Collins, W. Brown, and M. Sherman. GRASP: A Visualization Tool for Teaching Security Protocols, in CISSE, pages 75-81, 2006.

[21] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Relating strands and multiset rewriting for security protocol analysis. In Proc. of IEEE Computer Security Foundations Workshop, pages 35–51, 2000.

[22] D. Song. Athena: a new efficient automatic checker for security protocol analysis. In Proc. of IEEE Computer Security Foundations Workshop, pages 192–202, 1999.

[23] F. Thayer, J. Herzog, and J. Guttman. Strand spaces: Why is a security protocol correct? In Proc. of IEE Symp. on Security and Privacy, pages 160–171, 1998.

[24] A. Perrig and D. Song. Looking for diamonds in the desert: extending automatic protocol generation to three-party authentication and key agreement protocols, in Proc. of IEEE Computer Security Foundations Workshop, pages 64–76, 2000.

[25] C. Ho, C. Eastmana and R. Catramboneb, An investigation of 2D and 3D spatial and mathematical abilities, in Design Studies, 27(4), pages 505-524, 2006.