USING DEDUCTIVE KNOWLEDGE TO IMPROVE CRYPTOGRAPHIC PROTOCOL VERIFICATION

Zhiwei Li, Weichao Wang
Department of Software and Information Systems
University of North Carolina at Charlotte, NC USA
Email: zli19@uncc.edu and weichaowang@uncc.edu

***Abstract*—*An effective representation of principals' knowledge can greatly improve the efficiency of cryptographic protocol analysis. In this paper, we propose a mechanism to represent the deductive knowledge contained in a set of terms. Using Dolev-Yao model as an example, we design two algorithms to generate the knowledge representation and derive terms, respectively. We prove that using our knowledge representation, a principal can derive a term by using only constructive operations. To demonstrate the advantages of the proposed approach, we integrate it with Athena to build a new protocol verifier. The new approach will drastically reduce the number of states that are generated and analyzed during protocol verification. Experiments on several cryptographic protocols widely used for evaluating protocol verifiers demonstrate the improvements.*

I. Introduction

With the ever-increasing diversity of cryptographic protocols and security mechanisms that are adopted to protect our physical and cyber worlds, the efficiency and detection capabilities of protocol verifiers start to play an important role in security enforcement. In cryptographic protocols, principals (both legitimate and malicious) interact with each other by sending and receiving messages. Through these interactions, they learn new knowledge and match received information to existing knowledge to achieve security goals such as authentication and key establishment. Here we define knowledge as a group of terms [1] including information units such as principal names, random numbers, secret keys, and their derivation results. Therefore, deductive knowledge is an important component in protocol analysis since it defines the set of messages that can be understood and generated by protocol participants.

Because of its importance, deductive knowledge has been explicitly or implicitly used in security protocol analysis. For example, in Strand Space Model [2], initial knowledge of an attacker is specified by a set of penetrator strands. In constraint-based approaches [3]–[6], knowledge of intruders is represented by a set of constraints. In applied Pi Calculus [7], knowledge of intruders is organized into a frame, in which the order of received messages is preserved. Lowe [8] uses a set of terms to model intruder knowledge and uses them to identify guessing attacks.

One factor that restricts knowledge from being used more effectively during cryptographic protocol analysis is the way in which the information is managed. Existing approaches usually adopt one of the following two methods to represent deductive knowledge in security protocols. In method one, they consider only the initial knowledge of principals when the protocol starts. Since it fails to trace the dynamic changes of the information, it is almost impossible to integrate knowledge into the protocol verification procedures. In a more sophisticated method, protocol analyzers represent a principal's knowledge by constructing a union of all received and eavesdropped messages. Although this method captures all knowledge of a principal, few approaches intentionally use the information to assist protocol verification. In Section II we will show that using principals' knowledge we can greatly improve the efficiency of some protocol analysis tools.

In this paper, we propose to develop a new mechanism to represent deductive knowledge under Dolev-Yao model [9] and integrate it with strand-space protocol verifier Athena [10] to investigate the efficiency improvements in protocol analysis. We will design a decomposition algorithm to identify all information units in a principal's knowledge. We will prove that all messages in the principal's knowledge can be derived by applying only constructive operations such as concatenation and encryption to these units. In this way, we have an efficient mechanism to determine whether or not a message belongs to a principal's knowledge. To demonstrate the applications of the proposed approach, we extend the state structure of Athena with our knowledge representation mechanism and develop improved protocol verification algorithms. Schemes are designed to drastically reduce the number of states that are explored during security protocol analysis, thus improving the verification efficiency. We implement the approach with C++ and experiment it with six security protocols that are widely used for evaluating protocol verifiers. The experiment results show that our approach can greatly

This research is supported in part by NSF DUE 0754592.

improve the efficiency of Athena.

The remainder of this paper is organized as follows. In Section II, we introduce the architecture of Athena and the potential to improving its efficiency by integrating principals' knowledge. In Section III, we present the details of our approach. Specifically, we focus on the knowledge representation mechanism and reformed algorithms for protocol verification. In Section IV we present the experiment results to demonstrate the advantages of the proposed approach. In Section V we introduce the related work. Finally, in Section VI we conclude the paper and discuss future extensions.

II. Background of Athena

A. Notations and Attacker Model

In this paper, we assume that a principal's knowledge contains all terms (possible infinite) that can be derived by it. For example, if a principal knows a secret key k and the principal names B and C, the term $\{B, C\}_k$ is in its knowledge. We adopt the standard Dolev-Yao model to represent the principals' deduction capabilities. We assume that a protocol participant can resend a received message, or construct a new message by decomposing received packets into smaller units and reorganizing them.

Following the schemes in Athena, we assume that the deduction capabilities of principals in this approach can be summarized by the term operators illustrated in Figure 1. Therefore, a principal's knowledge can be represented as the closure of its initial information (e.g. private key and random numbers) and the received messages under these operators. We also use a "( )" to represent a set of terms.

Constructive operators:

Concatenation	$t_1 \quad t_2 \rightarrow \{t_1, t_2\}$
Encryption	$t_1 \quad t_2 \rightarrow \{t_1\}_{t_2}$
One way function	$t_1 \rightarrow h(t_1)$

Destructive operators:

Split	$\{t_1, t_2\} \rightarrow t_1$
	$\{t_1, t_2\} \rightarrow t_2$
Decryption	$\{t_1\}_{t_2} \quad t_2^- \rightarrow t_1$

Fig. 1. The Dolev-Yao deductive system.

Here we do not distinguish symmetric encryption from asymmetric methods. On the contrary, we use a uniform format key^- to represent the secret that is used to decrypt $\{m\}_{key}$. In symmetric encryption methods, key and key^- are the same. In asymmetric encryption, they represent the public/private key pair. Whether or not a principal can derive key^- from key is determined by their meanings and the protocol setup. The same deductive system has been adopted in [6].

B. Introduction to Athena

Song proposed Athena [11] to automate verification of cryptographic protocols. It adopted strand space model [2] to describe protocol execution procedures. In strand space model, a *strand* contains a finite sequence of nodes that describe the sending (+) and receiving (−) events happening at a principal. The nodes in the same strand and different strands are linked by the relationships $\Rightarrow$ and $\rightarrow$ respectively to demonstrate their happening orders. A *bundle* is a finite subgraph of strand spaces, in which every received message was sent by either a legitimate principal or an adversary. It can be viewed as a snapshot of the execution of a protocol. In Figure 2 we use the NSPK protocol [12] as an example to illustrate the terminology.

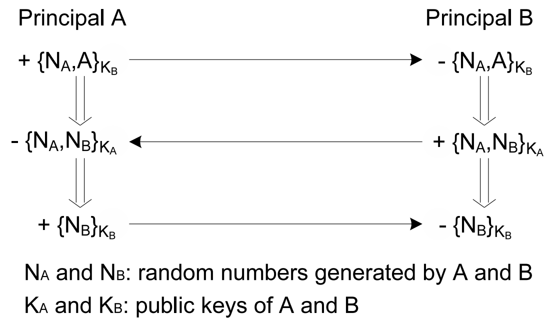


Fig. 2. NSPK protocol: A bundle.

In Athena, a *goal* represents a received message of a node and *goal-binding* is the causal relationship to locate the origin of this message. Those goals without bound sources are generally referred as unbound goals. Therefore, a state during protocol verification in Athena can be jointly determined by a group of strands and a set of unbound goals in them. Athena starts from an initial state and recursively generates new states to bind those unbound goals. This procedure will generate a growing state tree, in which new goals will be introduced. When there are no unbound goals in current state, Athena examines whether or not it contains strands from adversaries to evaluate the safety of the protocol.

C. Potential Improvements to Athena

Based on the previous description, we notice that Athena does not maintain a record of the dynamic knowledge of principals as the protocol proceeds. The lack of the information has several negative impacts on the efficiency of the verification procedure. First, since Athena does not know the latest knowledge of a principal, during the goal binding procedure we have to introduce new strands and new states to determine whether or not a message can be constructed by the principal. Previous studies [13] show that the number of explored states during verification has a direct impact on its efficiency.

Second, the lack of principal knowledge leads to a 'memoryless' verification procedure and will generate many unnecessary states in Athena. For example, the verification algorithm may have confirmed that when a principal's knowledge is a term set P, it cannot derive a message t. Therefore, it is safe for us to stop attempting to derive t in another state when the knowledge is $P' \subseteq P$. Our experiments show that in some security protocols, more than 7% unbound goals can be avoided by this technique.

Last but not least, Athena does not maintain the record when a goal is successfully bound through interactions among several principals. Therefore, we have to execute the same binding procedure every time that we want to bind the goal. The redundant efforts will also degrade the performance of the verification system.

III. Proposed Approach

To address these problems, we propose to develop a new mechanism to represent the knowledge of principals in cryptographic protocols. We will also integrate it with Athena to build a new protocol verifier. Below we focus on the knowledge learning and protocol verification algorithms. The system implementation and evaluation results will be discussed in Section IV.

A. Knowledge Representation and Deduction

The decision problem under Dolev-Yao model has been studied in [9], [14]. In this part, we first present a decomposition algorithm to identify the information units contained in a newly learned term and use them to extend the principal's knowledge. We will then prove that using these information units we can design an efficient algorithm to determine whether or not a term can be derived from the principal's knowledge.

Algorithm 1 : Derive $(P,\ t)$

```
Input: term set $P$, term $t$
1:  if ($t \in P$) then
2:    return true
3:  if ($t = \{t_1\}_{t_2}$ or $\{t_1, t_2\}$) then
4:    if (Derive($P, t_1$) and Derive($P,\ t_2$)) then
5:      return true
6:  if ($t = hash(t_1)$ and Derive($P,\ t_1$)) then
7:    return true
8:  return false
```

We first design Algorithm 1 to determine whether or not we can derive a term t from a term set P by applying only constructive operators such as concatenation, encryption, and hash functions to elements in P.

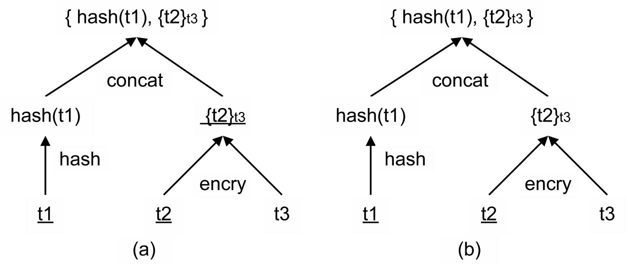


Fig. 3. Construction tree of $\{hash(t_1),\ \{t_2\}_{t_3}\}$. Terms in P are underlined. (a) Derive(P, t)= true. (b) Derive(P, t)=false.

The basic idea is to generate the construction tree of t and determine whether or not there is at least one element in P for every path linking a leaf node and the root of the tree. For example, when $t = \{hash(t_1),\ \{t_2\}_{t_3}\}$ and $P = (t_1, t_2, \{t_2\}_{t3})$, the construction tree of t is illustrated in Figure 3. For every path linking a leaf node and the root, we have at least one element in P. Therefore, the function Derive(P, t) will return "true". When we remove $\{t_2\}_{t_3}$ from P, the function will return "false". Since the function "Derive()" recursively applies decomposition operators to t, the recursion depth is restricted by the height of its construction tree and the function will terminate eventually.

Algorithm 2 : Learn $(P,\ t)$

```
Input: term set $P$, term $t$
1:  initial an empty Queue $Q$
2:  Insert($Q, t$)
3:  while(!Empty($Q$))
4:  {
5:    $x$ = popQueueHead($Q$)
6:    if ($x \notin P$) then
7:    {
8:      $P = P \cup x$
9:      if ($x = \{t_1, t_2\}$) then
10:     {
11:       Insert($Q, t_1$)
12:       Insert($Q, t_2$)
13:     }
14:     if ($x = \{t_1\}_{t_2}$ and Derive($P, t_2^-$)) then
15:       Insert($Q, t_1$)
16:     for every element $y$ in $P$
17:       if ($y = \{t_1\}_{t_2}$ and Derive($P, t_2^-$)
          and !Derive($P - x, t_2^-$)) then
18:         Insert($Q, t_1$)
19:   }
20: }
```

Now we use the "Derive()" function to design a learning algorithm. The algorithm will take a term set P and a term

t as inputs. It will identify the information units contained in t and use them to extend knowledge representation P.

The conditional statement in line (6) of the algorithm guarantees that every term will be learned only once. In line (8) to (15) of the algorithm, we will add term x into the knowledge representation and apply decomposition operators to x to enable the learning of its subterms. In line (16) to (18) we reexamine the terms in P. If the new term x enables us to decrypt some messages, we will insert the decryption results into Q for subsequent learning activities. In the whole algorithm, we apply only decomposition operators to terms in P. Since in Figure 3 we have shown that the height of construction trees of terms is limited, the $Learn()$ algorithm will terminate. Following the same analysis, we conclude that when we apply the $Learn()$ algorithm to a set of terms, the worst case of the complexity is proportional to the total number of nodes in these terms' construction trees. When we analyze real cryptographic protocols, the complexity is usually much lower because of the redundancy in messages.

As we introduce in Section I, the knowledge of a principal in a cryptographic protocol can be represented as the union of its initial knowledge, all received and eavesdropped messages, and newly generated information (e.g. random numbers). We represent the union as $R = (r_1, r2, \cdots, r_n)$. Below we show that if we start from an empty set P and call the $Learn()$ function to learn every term in R, the result set P will have the same closure as R under the Dolev-Yao model and hence represent the same knowledge as R. The analysis is as follows.

Since we call the $Learn()$ function for every term in R and a term will be added into P when it is first learned, we have $P \supseteq R$. Because of the monotonic property of the knowledge model, we know that the closure of R is a subset of that of P. On the other side, since every term that is added into P can be derived from R under Dolev-Yao rules, we know that the closure of P is a subset of that of R. Combining these results, we find that the output of the learning algorithm can be used to represent the knowledge of principals in cryptographic protocols.

Finally, we prove that using the learning outcome P, we can design an efficient mechanism to determine whether or not a term can be derived from a principal's knowledge under Dolev-Yao rules.

Proposition III.1. *Let R be a term set and P be the final outcome of applying the $Learn()$ algorithm to every term in R. For any term t that can be derived from R in a finite number of steps under Dolev-Yao deductive system, there exists a derivation procedure of t from P that uses only terms in P and constructive operators.*

Proof: We prove this proposition by designing a mechanism to remove those destructive operators from the derivation procedure. In Dolev-Yao deductive system, there are only two types of destructive rules: decryption and split. Since we assume that t is in the closure of R, we have t in the closure of P as well. We further assume that the derivation procedure of t from P contains destructive operations. These operations can be divided into three types based on how we get the destructed term, as illustrated in Figure 4. Type 1: the destructed term is the result of a constructive operation. Type 2: the destructed term is the result of another destructive operation. Type 3: there is no immediate operation before hand and the destructed term is in P. Below we will discuss these cases respectively.

For the first type, since the two pairs of constructive/destructive operations namely encryption/decryption and concatnation/split will cancel out each other, the destructive operation can be removed and we can directly use inputs to the constructive operation for future derivation. Figure 4.a presents an example. We can remove the encryption and decryption and directly use t_2.

For the second type, we seek to transform it to a type 1 or type 3 operation. We continue to move our investigation target to the operation immediately before as long as it is a destructive operator. Since the term that is decrypted or split is longer than the destruction result, this procedure will terminate and we will have a type 1 or type 3 case. After we remove that destructive operator, a type 2 case will become another type 1 or type 3 case. Figure 4.b presents an example.

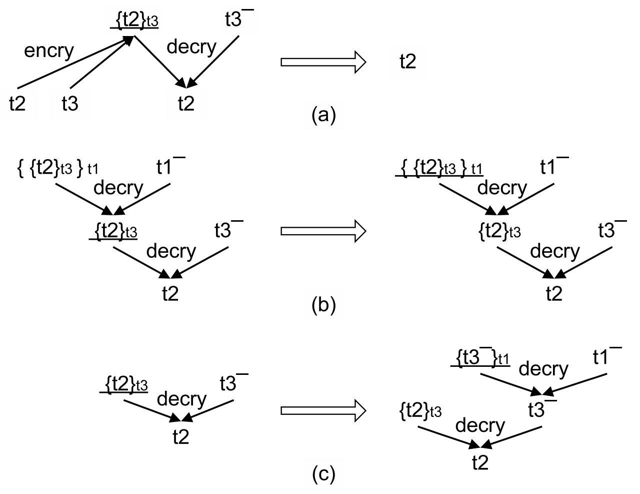


Fig. 4. Examples of three types of destruction. Underlined terms represent the destruction under investigation. (a) Type 1: remove the destruction. (b) Type 2: transfer to Type 1 or Type 3. (c) Type 3: one destruction leads to another destruction.

Finally, for the third type, the term that is decrypted or split is an element $p \in P$. If p has the format of $p = \{p_1, p_2\}$, we will be able to remove the split operator. As illustrated in line (9) to (13) of the learning algorithm, when we learn p, both p_1 and p_2 will be added into the queue Q. Therefore, we must have p_1 and p_2 in P and we do not need the split operation to get either of them.

If p has the format of $p = \{p_1\}_{p_2}$, we will need a decryption operator to get p_1 for subsequent derivation. Now we consider two possible conditions. (a) if $Derive(P, p_1)$ = true. We can remove this decryption operation and use only constructive operators to build p_1. (b) if $Derive(P, p_1)$ = false. We need to decrypt p with p_2^- to get p_1. Here we prove that $Derive(P, p_2^-)$= false. If $Derive(P, p_2^-)$= true, then either line 14 to 15 or line 17 to 18 of the learning algorithm will apply this result to $\{p_1\}_{p_2}$ and add p_1 into P. Since $Derive(P, p_2^-)$=false, there must be at least one destructive operation in its derivation procedure that cannot be removed. Based on previous analysis, it must be a type 3 decryption. Without losing generality, we call the destructed term $q = \{p_3\}_{p_4}$. Following the same analysis, we can prove that $Derive(P, p_4^-)$= false. In this way, we can locate another type 3 decryption of $\{p_5\}_{p_6}$ for the derivation of p_4^-. This procedure has to repeat forever and we will have a derivation with an infinite size. This is contradict to the assumption that t can be derived from R and P in a finite number of steps. Therefore, we conclude that condition (b) will not happen.

Combining cases 1, 2, and 3, we can remove all destructive operations in the derivation procedure of t from P. That implies $Derive(P, t)$ = true. $\square$

With this proposition proven, we can directly apply the $Derive()$ function to determine whether or not we can derive a term t from the learning result P. Since in $Derive()$ we use only decomposition operators, the recursion depth is proportional to the height of the construction tree of t.

B. Integrating Knowledge with Athena

In Section III.A, we introduce the algorithms to learn new knowledge and derive terms. In this part, we will investigate techniques to extend Athena with our proposed knowledge representation and design new protocol verification algorithms.

We propose to extend the state structure of Athena with the knowledge representation of principals. Therefore, the new states have a structure of $\langle S, G, (P)\rangle$ in which S contains the partially linked strands, G represents the set of unbound goals, and the set (P) contains the outputs of the $Learn()$ algorithm for every principal.

A principal has two methods to learn new knowledge. First, it can learn from received or eavesdropped messages in the network. Second, it can use the strands of the protocol to initiate connections to other parties to get new information (e.g. man-in-the-middle attack). These methods also explain the procedures through which a principal can bind an open goal. When a principal needs to bind a goal, it will first try to deduct it using its knowledge by calling the $Derive()$ function. If the derivation succeeds, the principal can generate the term without introducing new states. If the deduction fails, it will use the strands of the protocol to get the information from other parties. If this procedure succeeds, it will add the information into its knowledge so that the next time when the same goal is needed, it can avoid introducing new strands.

Although the scheme of term derivation through protocol strands is very similar to the goal binding procedure in Athena, our knowledge based approach is more efficient. In Athena, if the same goal is needed for multiple times, new strands will be introduced and new states will be added during the binding procedure of each time. In our approach, only the first time will introduce new strands. After that, the information will be added into the principal's knowledge and derivation through knowledge will be adopted in future bindings. In this way, we achieve 'learn once, use always' and can further reduce the number of states in verification.

Our new approach will verify a protocol as follows. We will first formulate the strands of principals into an initial state. Using the deduction rules and protocol strands, we try to bind those open goals and construct complete interactions among principals. Finally, we will identify those complete interactions that do not contain corresponding strands of other legitimate principals and generate the attack procedures.

In our new protocol verification algorithm, we will first try to bind a goal through knowledge derivation before introducing new strands. In this way, we guarantee that our approach will never have more states than Athena when they verify the same protocol. The conclusion will be supported by the experimental results in the next section.

IV. Experimental Results

To evaluate the proposed approach, we have implemented both Athena[1] and the knowledge based protocol verifier with C++. A snapshot of the system interface is illustrated in Figure 5. We use both tools to verify a set of well studied protocols. During the comparison, we find that in the original design of Athena, the protocol verifier will quit execution as soon as the first attack is located. Although this method will shorten the response time of the verifier, it will introduce some bias into the comparison results. When two verifiers adopt different schemes to choose the next state to examine during verification, they may generate different numbers of states when the first attack is located. To address this problem, we make some adjustments to Athena so that it will conduct a complete search to the whole state space. By examining

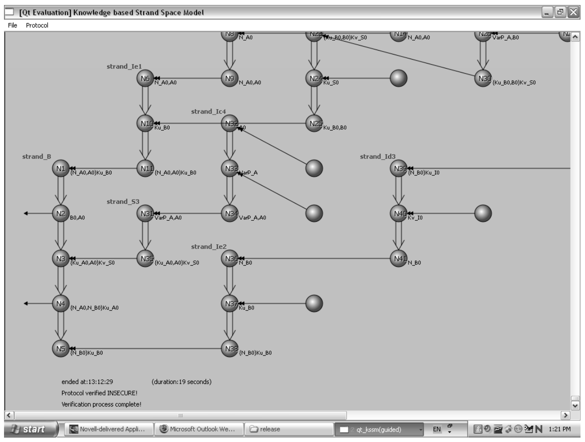
[1]To the best of our knowledge, Athena tool has never been released.

Protocol	Number of states explored			
	when the first attack is detected		the full state space search	
	knowledge based verifier	Athena	knowledge based verifier	Athena
WooLam	7	15	13	78
Neumann Stubblebine	97	135	117	2614
Needham Schroeder (TTP)	53	53	1411	2133
Needham Schroeder	42	68	55	112
Needham Schroeder Lowe	26 (secure)	43 (secure)	26 (secure)	43 (secure)
Otway Rees	2	7	28	326

TABLE I
EXPERIMENTAL RESULTS.

the whole state space, we avoid this bias and get a better understanding to the advantages of our approach.

Table I lists the number of states that are generated and examined by both tools. As shown in the table, in all tests the knowledge based algorithm reduces the number of states that are examined, especially in full search scenarios.



```
ended at:13:12:29        (duration:19 seconds)
Protocol verified INSECURE!
Verification process complete!
```

Fig. 5. Interface of knowledge based protocol verifier.

Further analysis shows that the generation and maintenance of the knowledge of principals introduce limited overhead to the system. While the analysis shows that the learning output of a group of terms may contain as many elements as the total number of nodes in their construction trees, during the verification of real protocols this number is usually much smaller. For example, Neumann-Stubblebine is probably the most complicated protocol studied in Table I and the learning output of the verifier contains fewer than 50 elements. As another example, the recursion depth of the $Derive()$ function can be as large as the height of the construction tree of the term. However, during the verification of real protocols, since there is a large amount of redundant information in the learning output, the recursion depth is usually much smaller.

V. RELATED WORK

Previous research efforts on applying knowledge to security protocol analysis can be divided into two groups: deducibility [15], [16] and indistinguishability [14].

The BAN [17] logic, proposed by Burrows, Abadi and Needham, is based on the deducibility notion of knowledge. It is probably the first extensively studied logic in protocol analysis based on knowledge. The agent's capability to synthesize messages is modeled by a set of inference rules. It is difficult to apply BAN logic to dynamically evolving knowledge to establish a generic model. There are many other logics designed for security protocol analysis [18], [19]. We find that most approaches using Dolev-Yao adversaries are based on deducibility notion.

Deducibility is one kind of algorithmic knowledge [20], in which "knowing what" can be determined by an algorithm. Halpern and Pucella have successfully used algorithmic knowledge to model several different adversaries [21]. Then Pucella proves that the decision problem in a general case is NP-complete [22]. Our work is inspired by their previous research on algorithmic knowledge.

The concept of indistinguishability comes from the classical possible-worlds approach to model knowledge [23], in which the actual world is considered to be one of many possible worlds. For example, in security protocol analysis, a principal cannot distinguish one random number from another if it has not seen these numbers before. Recently, Cohen and Dam [24] provide a generalized Kripke semantics for studying this type of knowledge in security protocols. They use a static equivalence [7] to capture the indistinguishability for agents.

Abadi and Cortier [14] examine the decidability of these two notions of knowledge by studying the underlying equational theories for deduction and static equivalence. This is especially important since the termination of knowledge analysis might not be guaranteed when the decidability result does not hold. Following this line of research, new decidability results are obtained for monoidal equational theories [25].

Though general enough, the notion of knowledge indistinguishability could not be easily applied to protocol verification. A Kripke style semantics usually suffers a *logical omniscience* problem [26]. Moreover, it is rather difficult to deal with explicit knowledge, which is widely used in most of the verification tools.

VI. Summary and future work

The lack of an effective representation of principals' knowledge restricts its potential to facilitating protocol analysis. In this paper, we propose a mechanism to represent the deductive knowledge under Dolev-Yao adversary model. We design two algorithms to generate the knowledge representation and derive terms based on the knowledge, respectively. We prove that both algorithms will terminate. At the same time, we prove that using our knowledge representation, a principal can derive a term by using only constructive operators.

To demonstrate the application and advantages of the proposed approach, we integrate it with Athena to build a new protocol verifier. The new approach will try to derive terms using principals' knowledge before introducing new states. In this way, our approach will generate and analyze fewer states during protocol verification and it is more efficient than Athena. Experiments on real protocols demonstrate the improvements.

The immediate extensions to our approach consist of the following aspects. First, we plan to extend our deduction systems so that principals' knowledge can be used to analyze more complicated protocols and detect more subtle attacks. Specifically, we will investigate the deduction systems such as Lowe's rule set. We will also study knowledge evolution under the operations such as xor and exponential-modular. Second, we plan to test our approach on cryptographic protocols deployed in real environments so that we have a better understanding of the applicability of the mechanism. We are especially interested in key establishment protocols for Internet and 802.11 wireless networks. These efforts will lead to a more effective knowledge representation and help us gain insight into protocol analysis.

References

[1] I. Bertolotti, L. Durante, R. Sisto, and A. Valenzano, "Efficient representation of the attacker's knowledge in cryptographic protocols analysis," *Form. Asp. Comput.*, vol. 20, no. 3, pp. 303–348, 2008.

[2] F. T. Fábegra, J. C. Herzog, and J. D. Guttman., "Strand spaces: Why is a security protocol correct?" in *Proc. of IEEE Symposium on Security and Privacy*, 1998, pp. 160–171.

[3] R. Corin and S. Etalle, "An improved constraint-based system for the verification of security protocols," in *Proc. of Symposium on Static Analysis*, 2002, pp. 326–341.

[4] R. Corin and A. Saptawijaya, "A logic for constraint-based security protocol analysis," in *Proc. of IEEE Symposium on Security and Privacy*, 2006, pp. 155–168.

[5] D. Kähler and R. Küsters, "Constraint solving for contract-signing protocols," *Proc. of Conference on Concurrency Theory*, pp. 233–247, 2005.

[6] J. Millen and V. Shmatikov, "Constraint solving for bounded-process cryptographic protocol analysis," in *Proc. of ACM conference on Computer and communications security*, 2001, pp. 166–175.

[7] M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *Proc. of ACM Symposium on Principles of Programming Languages*, 2001, pp. 104–115.

[8] G. Lowe, "Analysing protocols subject to guessing attacks," *Journal of Computer Security*, vol. 12, pp. 83–97, 2004.

[9] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[10] D. Song, S. Berezin, and A. Perrig, "Athena: A novel approach to efficient automatic security protocol analysis," *Journal of Computer Security*, vol. 9, no. 1/2, pp. 47–74, 2001.

[11] D. Song, "Athena: A new efficient automatic checker for security protocol analysis," in *Proc. of IEEE Computer Security Foundations Workshop*, 1999, pp. 192–202.

[12] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," Technical Report CSL-78-4, Xerox Palo Alto Research Center, Tech. Rep., 1978.

[13] C. J. Cremers, P. Lafourcade, and P. Nadeau, "Comparing state spaces in automatic security protocol analysis," *SICS LNCS special issue on Computer Security*, 2008.

[14] M. Abadi and V. Cortier, "Deciding knowledge in security protocols under equational theories," *Theor. Comput. Sci.*, vol. 367, no. 1, pp. 2–32, 2006.

[15] G. Lowe, "Breaking and fixing the needham-schroeder public-key protocol using fdr," in *TACAs '96: Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, 1996, pp. 147–166.

[16] L. Paulson, "The inductive approach to verifying cryptographic protocols," *Journal of Computer Security*, vol. 6, no. 1/2, pp. 85–128, 1998.

[17] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, 1990.

[18] L. Gong, R. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols," in *Proc. of IEEE Symposium on Security and Privacy*, 1990, pp. 238–248.

[19] S. Stubblebine and R. Wright, "An authentication logic supporting synchronization, revocation, and recency," in *Proc. of ACM conference on Computer and communications security*, 1996, pp. 95–105.

[20] J. Halpern, Y. Moses, and M. Vardi, "Algorithmic knowledge," in *Proc. of 5th conference on Theoretical Aspects of Reasoning about Knowledge*, 1994, pp. 255–266.

[21] J. Halpern and R. Pucella, "Modeling adversaries in a logic for security protocol analysis," in *In Formal Aspects of Security, First International Conference, FASec 2002*, 2002, pp. 115–132.

[22] R. Pucella, "Deductive algorithmic knowledge," *Journal of Logic and Computation*, vol. 16, no. 2, pp. 287–304, 2006.

[23] J. Halpern, "Reasoning about knowledge: a survey," *Handbook of logic in artificial intelligence and logic programming*, vol. 4, pp. 1–34, 1995.

[24] M. Cohen and M. Dam, "A complete axiomatization of knowledge and cryptography," in *LICS '07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, 2007, pp. 77–88.

[25] V. Cortier and S. Delaune, "Deciding knowledge in security protocols for monoidal equational theories," in *Proc. of International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, 2007, pp. 192–210.

[26] R. Fagin, J. Halpern, Y. Moses, and M. Vardi, *Reasoning about Knowledge*. The MIT Press, 1995.