Research on Security Protocol Analysis Tool SmartVerif

Siqi Lu¹, ²*, Qingdi Han³, Xuyang Miao¹, ² and Yubo Liu¹, ²

¹ State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, Henan, 450000, China
² Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, Henan, 450000, China
³ Department of Information Science and Engineering, Ocean University of China, Qingdao, Shandong, 266100, China

*Corresponding author’s e-mail: 080lusiqi@sina.com

Abstract. Security protocols have been designed to protect the security of the network. However, many security protocols cannot guarantee absolute security in real applications. Therefore, security tests of the network protocol become particularly important. In this paper, firstly, we introduce SmartVerif, which is the first formal analysis tool to automatically verify the security of protocols through dynamic strategies. And then, we use SmartVerif to verify the pseudo-randomness of the encapsulated key of the Two-Pass AKE protocol, which was proposed by Liu’s in ASIACRYPT in 2020. Finally, we summary our work and show some limitations of SmartVerif. At the same time, we also point out the direction for future improvement of SmartVerif.

1. Introduction

The security analysis of the network protocol can find out the vulnerabilities and defects in the network protocol, and discover the potential attacks. The earliest way used to analyze the network protocol is an informal method, namely, attack verification method. It simulates an intruder and uses a variety of different attack methods to attack the protocol. If the attack is successful, then the vulnerabilities in the network protocol are found. However, this method has great limitations, because it can only detect whether there are known vulnerabilities in the protocol, and nothing to do with unknown attacks.

Compared with the informal method, the formal method is more popular. The formal analysis of security protocols began with the Dolev-Yao model proposed by Dolev and Yao[1], which has become the theoretical basis for most of the formal analysis of protocols. Its essence is to abstract concepts and methods into mathematical models with symbols and various rules of using these symbols, and then study the mathematical model through program derivation and calculation, and finally explain the rules of its internal change. The main feature of the formal method is that it can accurately describe the process of protocol execution and the security attributes need to be satisfied, and infer whether these security attributes are achieved. Compared with informal analysis methods, formal analysis methods can comprehensively and deeply detect the subtle security vulnerabilities and defects in the protocol.

SmartVerif[2] is a popular tool for formal analysis of security protocols, and many scholars have used it to analyze a series of security protocols, which has shown its effectiveness in formal analysis of security protocols. In this paper, we introduce SmartVerif in briefly. And then we use this tool to analyze the Two-Pass AKE protocol[3].
The rest of the paper is organized as follows. The second section of this article mainly introduces the SmartVerif; the third section mainly analyzes the Two-Pass AKE protocol process and models, and verifies the protocol; the last section summarizes and prospects the work of this paper.

2. Introduction of SmartVerif
The development of the SmartVerif tool is based on Tamarin-prover\[4,5\] and some modifications have been made to it\[6\]. SmartVerif uses a dynamic strategy internally, which can be used to intelligently search for the proof path. Different from the non-trivial and error-prone design of existing static strategies, the design of SmartVerif's dynamic strategy is simple and flexible: it can automatically optimize itself according to the security protocol without any manual intervention. By optimizing the strategy, SmartVerif can locate and prove to support the lemma, thereby increasing the verification success rate.

The design idea of this strategy is that when a random strategy is given, the probability that a node supporting the lemma is on the wrong proof path is low. The verification here can be simply seen as the process of path search in the tree: each node represents a state, used to prove that the lemma on the root or path is correct, only when each node on the path represents the status is correct to show that this is a complete and correct path.

During the operation of SmartVerif, if the currently selected path is incorrectly estimated, the dynamic strategy will optimize it, if the path proves to be incomplete, the strategy will search for other complete paths. Its initialization does not require any manual intervention the initial strategy is completely random. After fully optimizing the strategy, the next search node will be selected intelligently.

SmartVerif's dynamic strategy is based on this. When a random strategy is given, it means that a node that supports the lemma has a low probability of being on the wrong path. SmartVerif introduces Deep Q Network (DQN)\[7\], a reinforcement learning algorithm, into the verification. DQN uses an empirical replay mechanism\[8\] according to the historical error path update strategy, which randomly selects the previous conversion and smoothly train distribution on the error path. Therefore, the optimization strategy often selects the nodes that support the lemma among the candidate nodes, and the probability of successful verification is higher.

The workflow of the SmartVerif is as follows:

Step 1: Initialize Deep Q Network with a purely random strategy, take multiple candidates as input, and randomly select a candidate with the same probability as output.

Step 2: SmartVerif uses the current strategy for proof search. It is executed in parallel through multiple threads. In each thread, Tamarin-prover is used as the basis to generate a proof path, in which each node on the path is selected according to the strategy.

Step 3: If the path generated in the second step is correct and complete, SmartVerif will terminate and output the path as the result.

Step 4: If all the path estimates generated in the second step are incorrect, SmartVerif will start a new cycle, use DQN to optimize the path and update the strategy based on the proof path.

Step 5: Go to Step 2.

3. Formal analysis of Two-Pass AKE Protocol
In this section, we first introduce the Two-Pass AKE protocol, and then model and verify the protocol by using the tool SmartVerif.

3.1. Introduction of Two-Pass AKE Protocol
The Two-Pass AKE protocol was proposed by Shengli Liu and others on ASIACRYPT in 2020. It is a key exchange protocol that uses an explicit authentication method to detect whether the protocol has been replayed during operation. If an attack is found, then the agreement no longer continues to run. The agreement process is as shown in figure 1\[2\]. In the process, AKE.Setup: input parameters are $l^A$, etc.
output common parameters are $PP_{AKE} = (PP_{SIG}, PP_{KEM})$. AKE.Gen: input parameters are $PP_{AKE}$ and $P_i$, output key pair are $(vk_i, sk_i)$. The interaction process of the protocol is as follows:

1. $P_i$ generates a key pair $(pk_{KEM}, sk_{KEM})$ through $KEM.Gen(pp_{KEM})$, records $(m_i = (P_i, P_j, pk_{KEM}))$, signs $sk_i$ and $m_i$ through $\sigma = SIG.Sign(sk_i, m_i)$, and sends $(m_i, \sigma_i)$ to $P_j$.

2. $P_j$ gets $(m_i, \sigma_i)$, verifies $vk_i$ with $\sigma_i$, if everything goes well, then $m_i$ will be a valid message. Otherwise, returns $(reject, \emptyset)$. When $m_i$ is valid, $P_j$ stores $(m_i, \sigma_i)$, encapsulates key $K$ through $(vk_i, sk_i)$, and sends $(m_i, \sigma_i)$ to $P_i$. $P_i$ accepts $K$ as the session key with $P_j$ by returning $(accept, K)$.

3. After $P_i$ getting $(m_i, \sigma_i)$, verifies whether $(m_i, \sigma_i)$ is a valid message signature pair. If all goes well, $P_i$ takes $m_i$ as a valid message and decapsulates the ciphertext $C$ in $m_i$ to get $K = KEM.Decap(sk_{KEM}, C)$. $P_i$ will accept $K$ as the session key with $P_j$, and return $(accept, K)$.

If $m_i$ is invalid, $P_i$ will return $(reject, \emptyset)$.

![Figure 1. Two-Pass AKE protocol process.](image-url)

### 3.2. Modeling and verifying the Two-Pass AKE Protocol

We utilize the SmartVerif to analyze the Two-Pass AKE protocol in three steps. The first step is to model the Two-pass AKE protocol, and then model lemma. The final step is to analyze the protocol with the SmartVerify and derive the analysis results.

1. **Protocol modeling.** The process of modeling the Two-Pass AKE protocol is shown in figure 2.

2. **Lemma modeling.** The process of modeling the protocol is shown in figure 3. KEM’s IND- mCPA security considers the pseudo-randomness of multiple encapsulation keys $\{K_i | (K, C) \leftarrow Encap(pk_i)\}$, where $\{(pk_i, C)\}$ is the corresponding public key and challenge ciphertext. This protocol now considers
a stronger attacks \textit{IND-mCPA}, which allows the adversary to know any \((pk_i, C)\), and allows the adversary to calculate the key \(K\) obtained from \(C\) and \(sk_i\). In addition, it also requires pseudo-randomness of unknown keys.

(3) Analysis result: The analysis result of the Two-Pass AKE protocol is shown in figure 4.

```
theory Two_pass_AKE
begin
  functions: m1/3, m2/3, f1/1, f2/1, f3/1, pk/1
  builtins: signing

  rule key_KEM:
  [ Fr(~skKEM) ]
  -->
  [ !Ltk0( $I, ~skKEM ), !Pk0( $I, pk(~skKEM) ), Out( pk(~skKEM)) ]

  rule key_i:
  [ Fr(~ski) ]
  -->
  [ !Ltk( $I, ~ski ), !Pk( $I, pk(~ski) ), Out( pk(~ski)) ]

  rule key_j:
  [ Fr(~skj) ]
  -->
  [ !Ltk( $I, ~skj ), !Pk( $I, pk(~skj) ), Out( pk(~skj)) ]

  rule I_1:
  let
  pkKEM = pk(skKEM)
  mm1 = m1($I, $J, pkKEM)
  sigma1 = sign(mm1, ski)
  in
  [ !Ltk($I, ski), !Ltk0( $I, skKEM ) ]
  -->
  [ !Pk($I, pkKEM), Out(<mm1, sigma1>) ]

  rule J_1:
  let pkKEM = pk(skKEM)
pki = pk(ski)K = f1(pkKEM)C = f2(pkKEM)mm2 = m2($I, $J, C)
sigma2 = sign(mm2, skj)
in
  [ In(<mm1, sigma1>), !Ltk($I, skj)), !Pk0( $I, pk(skKEM)), Verify(sigma1, mm1, pki) ]
  -->
  [ Out(<mm2, sigma2>) ]

  rule I_2:
  let pkKEM = pk(skKEM)
pkj = pk(skj)K1 = f3(pkKEM, C)
in
  [ Int(<mm2, sigma2>), !Pk0( $I, pk(skKEM)), Verify(sigma2, <mm1,mm2>, pkj) ]
  -->
  [ Out(K1) ]

  rule Sessk_reveal:
  [ !Sessk( K1 ) ] --> RevealSessk( K1 ) ]
  -->

Figure 2. Model the Two-pass AKE protocol.
```
lemma known_encap:
"All K1 pki C #i #j. Accept(pki, C) @i & RevealSessk( K1 ) @j==> F"

Figure 3. Model the lemma.

summary of summaries:
analyzed: Two_pass_AKE.spthy
known_encap (all-traces): verified (6 steps)

Figure 4. The analysis result of the Two-Pass AKE protocol.

We can see that the security attributes we modeled before have been successfully verified. Using the interactive mode yields SmartVerif’s automated proof process for security properties, as shown in figure 5.

Figure 5. Two-Pass AKE Protocol Lemma

The result shows that the modeled security protocol is effective and has been verified.

In summary, we can obtain that the Two-pass AKE Protocol can meet the security requirements of revealIND mCPA, even if the opponent knows (pk_i, C), the final key cannot be obtained.

4. Conclusion and Future
This paper studies the automated analysis tool SmartVerif for security protocols, models the Two-pass AKE protocol, and analyzes the Two-pass AKE protocol. The security attributes in the protocol are analyzed, and the IND-mCPA reveals security attributes of the Two-Pass AKE protocol are successfully verified.

SmartVerif currently has some limitations: Frist, it is necessary to train an independent DQN for each research protocol, and there is no general pre-training neural network. Second, it is also necessary to retrain the DQN when verifying the new security attributes of the protocol. In actual use, the DQN needs to be retrained after modifying the protocol or attribute specification.

The future work of SmartVerif is mainly in the following aspects:
Use a hybrid strategy. Since the initial strategy of SmartVerif is purely random, if the initial strategy uses a static strategy, it can be optimized to fewer epochs.

Scalability. Such as automatic formal verification of software or systems based on first-order logic or higher-order logic. They are very similar in form and can be transformed into a path search problem.

Efficiency. Designing a common network that can verify all protocols can improve the efficiency and performance of SmartVerif.

Acknowledgments
The work was supported by National Natural Science Foundation of China(61872381).

References
[1] Dolev D and Yao C C. (1981) On the security of public key protocols. In: Symposium on Foundations of Computer Science. IEEE.
[2] Xiong Y, Su C, Huang W, Miao F, Wang W, and Ouyang H. (2020) SmartVerif: Push the Limit of Automation Capability of Verifying Security Protocols by Dynamic Strategies. In: School of Computer Science and Technology, University of Science and Technology of China. USENIX Security Symposium. pp. 253–270.
[3] Liu X, Liu S, Gu D, and Weng J. (2020) Two-Pass Authenticated Key Exchange with Explicit Authentication and Tight Security. In: Shanghai Jiao Tong University, Jinan University of China. ASIACRYPT.
[4] Steve K and Robert K. (2014) Automated analysis of security protocols with global state. In 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, pp. 163–178.
[5] Simon M, Benedikt S, Cas C, and David A. B. (2013) The TAMARIN prover for the symbolic analysis of security protocols. In Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19. pp. 696–701.
[6] Benedikt S, Simon M, Cas J. F. C and David A. B. (2012) Automated analysis of diffie-hellman protocols and advanced security properties. In: In 25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27. pp. 78–94.
[7] Volodymyr M, Koray K, David S, Andrei A. R, Joel V, Marc G. B, Alex G, Martin A. R, Andreas F, Georg O, Stig P, Charles B, Amir S, Ioannis A, Helen K, Dharshan K, Daan W, Shane L, and Demis H. (2015) Human-level control through deep reinforcement learning. In: Nature, 518(7540). pp. 529–533.
[8] Lin L. (1993) Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.