SANS Security Essentials GSEC Practical Assignment

A Look at Automatic Protocol Generation & Security Protocols

By Boris W. Vassall

July 16, 2001
(Original Submission)

Table of Contents

Introduction

This paper will attempt to describe automatic protocol generation, and security protocols. Automatic Protocol Generation, APG for short, is a mechanism to generate security protocols automatically. This is accomplished by having the designer or engineer input a set of security system requirements and properties that dynamically produces a security protocol that best meets the criteria. [APDS00] The system requirements for input are defined as a metric function, which defines the cost or overhead of the protocol primitives, which defines an ordering over protocols with respect to the metric function. Based on this ordering, APG investigates the protocol space and outputs the correct protocol, which has minimal cost with respect to the metric function. The protocol also satisfies the security properties and system requirements.

The advantage of the Automatic Protocol Generation (APG) approach over the current protocol design process is that, it is fully automatic. The designer inputs the properties and system requirement's which result in a security protocol or output. This is by far, a better process than creating the security protocol manually. The protocols generated by APG have a higher level of confidence. This high level of confidence is a result of being able to verify with a powerful protocol analyzer. Another advantage of APG is that since with respect to the order of increasing cost on the Metric Function, APG searches through the protocol space and generates correct protocols with minimal cost that are in line with the system requirements. Further advantages would be that APG is very flexible in the sense that it can handle different security properties and system requirements.

Discussion on Security Protocols

Security Protocols play a pivotal role in the overall scheme of e-commerce and the Internet. Every day new attacks, compromises, and viruses threat the ability to securely and effectively transact data between "fellow-users" on the Internet. The role of a Security Protocol

is to utilize cryptographic building blocks to achieve security goals such as authentication, confidentiality, and integrity. Newer applications and systems may require existing security protocols to be revised or updated to meet more modern system requirements. Security protocol development is a delicate task, and history has shown that security protocol development is very challenging. [BAN89, Low96]
The current security protocol design process is arguably flawed for the following reasons:

- It's Error-prone. Security protocols should be intricate because attackers are powerful.
- Manually designed protocols are flawed because they contain undocumented assumptions, which is a result of the lack of formalism and mechanical assistance.
- The protocol designer lacks the expertise and experience and is more than likely to develop a non-efficient protocol that is flawed fundamentally.
- Non-optimal, the designer may include unnecessary operations. Conservative designers may include unnecessary operations just to play it safe.
- Inefficient and Expensive. The design process can be slow, and can potentially become the bottleneck to the project. High costs can be incurred due to redesign, update plans, or liability claims.

APG is a mechanism to address the above shortcomings of manually developed security protocols. APG allows the designer to specify the desired security properties such as authentication and secrecy, system requirements, and low bandwidth. System requirements are defined as symmetric or asymmetric encryption/decryption.

(Symmetric encryption is defined as the process of encrypting using a single key to both encrypt and decrypt data.)

(Asymmetric encryption is defined as the process of encrypting using two keys one for encrypting and the other for decrypting.)

The protocol generator after receiving the required inputs it generates a candidate security protocol, which satisfies the system requirements. In the final stage a protocol screener analyzes the candidate protocols, ignores the flawed protocols, and creates the correct protocols that address the desired security properties. The benefits of this approach is that it provides the following:

- Automatic, the designer specifies the security parameters and properties but the remaining process is automatic.
- Provides a High Confidence level. There are no hidden assumptions as is the case in the manual protocol development process.
- The protocol screener is powerful enough to generate a proof if the protocol is correct or a counterexample. Thus providing further confidence in the process.
- High Quality, the user defined requirements include a metric function which specifies cost overhead of a protocol.
- Flexibility, this mechanism works for different security properties, system requirements, and attacker models.

Framework Requirements for APG

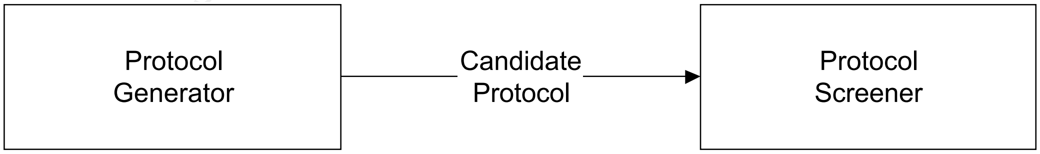
From a high perspective, APG is composed of an automatic protocol generator and an automatic protocol screener as depicted by our figure illustrated below. The process of APG has four stages.

Stage 1: The protocol designer must define the security properties and system requirements for input.

Stage 2: The protocol generator searches the protocol space and generates candidate protocols that satisfy the system requirements in stage 1.

Stage 3: The protocol screener analyzes the candidate protocols.

Stage 4: The flawed protocols are discarded and the correct ones that satisfy the security properties are outputted.



Specification for Input

A specification language was developed to define the security properties necessary for two parties to talk to each other. The properties are security, authentication, secrecy and other properties related to e-commerce. The system requirement is specified as a metric function and is a part of the initial setup. So what does the initial system configuration define? It defines which cryptographic primitives are available to the principals and what keys each principal possess. What does this mean? Well let's take for example a asymmetric-key such as PKI. All the parties know their own private key and the public keys of the other principals. In a symmetric-key environment the principals have shared secret keys. Hybrid systems are also possible.

The Metric function equates to the cost or the overhead of the protocol. For example, in metric design it's possible to make the metric correspond to the time overhead of the protocol. To further demonstrate this example, let's take for instance smart-card technology. In a smart-card system, encryption can be very fast however the bandwidth between the card and the reader may be slow, in which case the metric function specifies a low cost for encryption, but a high cost for sending and receiving messages. The metric function increases monotonically as the protocol complexity is increased. This requirement is necessary during the protocol generation phase, where protocols up to the maximum cost threshold are generated. To further clarify and simplify, the role of the metric function is to define an order among the protocols generated by the protocol generator. When given a specification of security properties and system requirements, the protocol is *optimal* if it has the lowest cost-value with respect to the metric function.

Protocol Generator

An obvious question produced out of the preceding section is, what is the role of the Protocol Generator. The Protocol Generator's function is to generate candidate protocols that satisfy the particular system requirements. When we observe closely the protocol space that the protocol generator works with. One finds that that space is virtually infinite. This poses another challenge, how do we limit the number of potential protocol candidates without omitting any potential optimal protocols. The answer to this question lies in a process called *iterative deepening.* This process, plain and simply put, is a search algorithm. The way this algorithm works is, that a cost threshold of protocols is set in each iteration. Then a search is done in the protocol space to generate all the protocols below the given threshold. Next, after the protocols are sorted by their costs, the protocol screener tests them. If a protocol satisfies the desired properties (Which means that it's cost is minimal) the generation process can stop. Otherwise, we increase the cost threshold and generate more protocols. To also aide in the process, a reduction technique is used to prune invalid candidate protocols early before they are passed on to the Protocol Screener. Many of the generated protocols include severe security flaws, which can be detected by a verification algorithm. A pruning algorithm is used to discard most severely flawed protocols. The Protocol Screener uses a verification condition.

Protocol Screener

The role of the Protocol Screener is given a candidate protocol, the screener must examine the protocol and tell whether it's verifiable or not. The protocol screener is reliable when it claims that a protocol satisfies certain security properties. Since the protocol generator produces thousands of protocols, the protocol screener is required to be very efficient in its task to find the optimal protocol in a reasonable amount of time.

So the next logical question would be, how does the Protocol Screener handle the task of verification of the protocols it receives from the generator? To answer this question we must look at a few verification techniques. Basically there are two types of verification techniques. The two techniques are *Automatic & Semiautomatic protocol analysis*.
Semiautomatic protocol analysis tools are NRL Analyzer [Mea94], the Interrogator Model [Mil95], FDR [Low96], and Brutus [CJM98]. Althena however, is an automatic protocol analysis tool that is most preferred automatic protocol analyzer because of the following reasons;

- Althena has the ability to analyze protocol executions that have any arbitrary configurations. Many existing automatic analyzer tools can only reason about finite state space. When Althena terminates, it proves that a protocol satisfies its specified properties under any arbitrary protocol configuration, or it demonstrates a counterexample if the property does not hold.

- Althena can exploit state space reduction techniques, which as a result provides a highly reduced state space.

- Althena provides a proof if a protocol satisfies a given property.

Protocol Representation

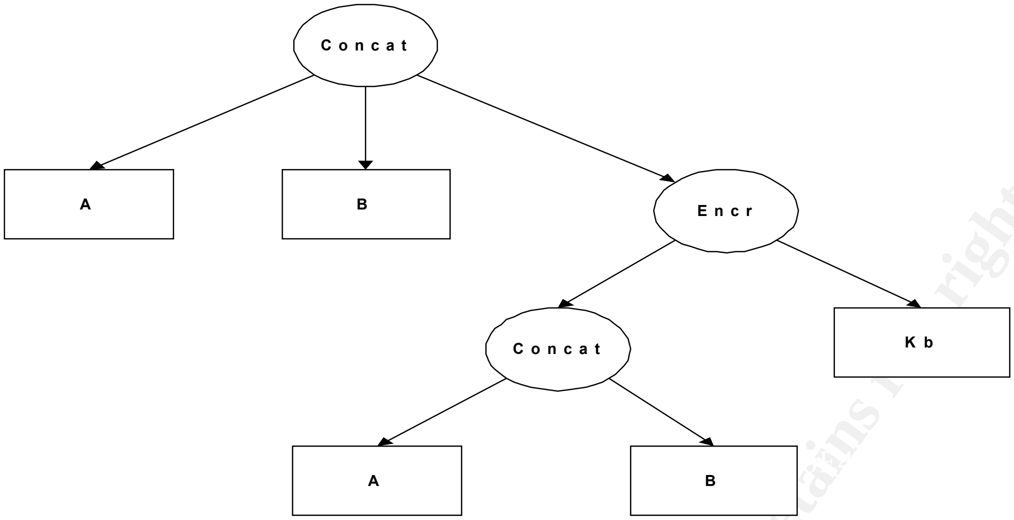
A protocol represents the sequence of actions of two communicating parties. The actions include sending and receiving messages. These messages are defined by the grammar listed below, and can easily be extended as needed. This also helps support the argument of flexibility in AGP.

```
Message ::= Atomic |Encrypted |Concatenated
Atomic ::= Principalname |Nonce |Key
Encrypted ::= (Message, Key)
Key ::=PublicKey | PrivateKey| SymmetricKey
Concatenated ::= Message List
Message List ::= Message | Message, Message List
```

A tree can also represent each Message, with the atomic messages as leaves and operations as intermediate nodes. In the figure below we illustrate an example for the message ***A,B, {A,B}kb.*** The *depth* of a message is defined as the depth of the tree representing the message. In the example below the message depth is 4.

Notation

A, B are the principals
N_A is a nonce generated by A
K_A denotes A's public key
K_A^{-1} denotes A's private key



For further information: http://paris.cs.berkeley.edu/~perrig/projects/protgen/node6.html

Case Study Discussion

For the purposes of keep this paper brief we have elected just to give a brief summary of the case study found [APDS00]. However, I recommend further reading of this case study, because helps to simplify the complexities of the process.

1. Assumptions
    1. Message components are typed
    2. No redundant message components in the concatenation
    3. No initial keys are sent in a message.
    4. The initiator's name needs to be in the first message in a format understandable to the responder.
    5. We don't consider permutations of the message components of a concatenated message.

2. A pruning Algorithm is developed for each security property, which prunes the majority of the protocols.

3. For impersonation attempts, we use two intruders to attack each protocol. The Intruder I_1 tries to impersonate the initiator. A, and

the other intruder I_r, tries to impersonate the responder B.

4. If the protocol screener outputs a flawed protocol, the automatic protocol generation is not trustworthy. The screener has to be efficient because the generator could generate thousands of protocols.

5. A simple linear metric function is used in the experiment. Each operation has a unit-cost. The cost value of a protocol is the sum of the costs of all the protocol operations and components.

Example of Symmetric-Key mutual authentication protocols:

A -> B: N_A, A
B -> A: { $N_{A,}$ $N_{B,}$ A} $_{KAB}$
A -> B: N_B

A -> B: N_A, A
B -> A: { $N_{A,}$ $N_{B,}$ B} $_{KAB}$
A -> B: N_B

Example of ISO Symmetric-Key three-pass Mutual Authentication Protocol:

A ->B: N_A, A
B ->A: { $N_{A,}$ $N_{B,}$ B} $_{KAB}$
A ->B: { $N_{A,}$ N_B} $_{KAB}$

Example of Asymmetric-key mutual authentication protocols:
(The following protocol is the same as the fixed version of Needham-Schroeder protocol[Low96])

A -> B: { N_A, A} $_{KB}$
B -> A: { $N_{A,}$ $N_{B,}$ B} $_{KA}$
A -> B: N_B

Closing Discussion

[APDS00] With a user-defined specification of security properties and system requirements, including a system metric function, APG generates minimal protocols that satisfy the specified security properties and system requirements, minimal with respect to the metric function. This strategy is a significant improvement over the current protocol design process, because it is more reliable, efficient, and produces protocols that are comparable to the given system requirements.

After further reviewing the proof of concept case study generated by A. Perrig and D. Song [APDS00] one can conclude that this study supports the theory that APG is a viable option. However, the case study mainly covers the authentication security property. There are other interesting security properties such as those related to e-commerce, such as atomicity. The process needs to be extended to include these properties. Atomicity is the process of performing all of a series of instructions or none at all. (Further clarification can be found [Tyger01]).

In the case study performed by A. Perrig and D. Song [APDS00], perfect encryption was an assumption. This assumption states that a cipher text can only be decrypted if the decryption key is present, and similarly, a cipher text can only be produced if the encryption key is present. Researchers are still exploring protocols, which are resistant to attacks such as dictionary attacks. However it is equally important to try and strengthen the attacker model to produce stronger protocols.

References

[APDS00] A. Perrig, D. Song, On a First Step to the Automatic Generation of Security Protocols. Computer Science Department, University of California, Berkeley http://paris.cs.berkeley.edu/~perrig/projects/protgen/node2.html

[Song99] D. Xiaodong Song, A new Efficient Automatic Checker for Security Protocol Analysis. Computer Science Department, Carnegie Mellon University 1999. http://citeseer.nj.nec.com/211930.html

[PS00] Looking for Diamonds in the Desert --- Extending Automatic.. - Perrig, Song (2000) (Correct)...for Diamonds in the Desert Extending Automatic Protocol Generation to Three-Party Authentication and Key Agreement Protocols Adrian Perrig http://citeseer.nj.nec.com/perrig00looking.html

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Tools and Algorithms for the Construction and Analysis of Systems, volume 1055 of Lecture Notes in Computer Science, pages 147-166. Springer-Verlag, 1996.

[Tygar01]
Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce(J.D.Tygar, 1998) [New] 06/05/01
http://www.vldb.org/conf/1998/p001.pdf

[BAN89]
M. Burrows, M. Abadi, and R. Needham. A logic of authentication. Technical Report 39, DEC Systems Research Center, February 1989.

[CGP99]
Edmund Clarke, Orna Grumberg, and Doron Peled. Model Checking. MIT Press, 1999.

[CJM98] E.M. Clarke, S. Jha, and W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In Proceedings of the IFIP Working Conference on Programming Concepts and Methods (PROCOMET), 1998.

[DY89]
D. Dolev and A. Yao. On the security of public key protocols. IEEE Transactions on Information Theory, 29(2):198-208, March 1989.

[HT96]
N. Heintze and J. Tygar. A model for secure protocols and their compositions. IEEE Transactions on Software Engineering, 22(1):16-30, January 1996.

[Int93]
International Standards Organization. Information Technology - Security techniques -- Entity Authentication Mechanisms Part 3: Entity authentication using symmetric techniques, 1993. ISO/IEC 9798.

[Low96]
G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Tools and Algorithms for the Construction and Analysis of Systems, volume 1055 of Lecture Notes in Computer Science, pages 147-166. Springer-Verlag, 1996.

[Low97]
G. Lowe. A hierarchy of authentication specifications. In Proceedings of the 1997 IEEE Computer Society Symposium on Research in Security and Privacy, pages 31-43, 1997.

[Mea94]
C. Meadows. A model of computation for the NRL protocol analyzer. In Proceedings of the 1994 Computer Security Foundations Workshop. IEEE Computer Society Press, June 1994.

[Mea95]
C. Meadows. Formal verification of cryptographic protocols: A survey. In Advances in Cryptology - Asiacrypt '94, volume 917 of Lecture Notes in Computer Science, pages 133-150. Springer-Verlag, 1995.

[Mil95]
J. Millen. The Interrogator model. In Proceedings of the 1995 IEEE Symposium on Security and Privacy, pages 251-260. IEEE Computer Society Press, 1995.

[MMS97]
J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using mur&phiv;. In Proceedings of the 1997 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 1997.

[RN95]
Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall Series in Artificial Intelligence, 1995.

[Son99]
Dawn Song. Athena: An automatic checker for security protocol analysis. In Proceedings of the 12th Computer Science Foundation Workshop, 1999.

[THG98]
F.Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In Proceedings of 1998 IEEE Symposium on Security and Privacy, 1998.

[WL93]
T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In Proceedings of the IEEE Symposium on Research in Security and Privacy, 1993.