A Look at Automatic Protocol Generation & Security Protocols

Boris Vassall
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 ................................................................. 3
Discussion on Security Protocols ........................................ 3
Framework Requirements for APG ....................................... 5

**Specification for Input** ............................................... 5
**Protocol Generator** ...................................................... 6
**Protocol Screener** ..................................................... 7
**Protocol Representation** ............................................... 8
Notation ............................................................................ 8

Case Study Discussion ...................................................... 9
Closing Discussion .......................................................... 11
References ........................................................................ 12
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.

![Diagram of the APG process]

**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 an 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.

\[
\text{Message ::= Atomic | Encrypted | Concatenated}
\]
\[
\text{Atomic ::= Principalname | Nonce | Key}
\]
\[
\text{Encrypted ::= (Message, Key)}
\]
\[
\text{Key ::= PublicKey | PrivateKey | SymmetricKey}
\]
\[
\text{Concatenated ::= Message List}
\]
\[
\text{Message List ::= Message | Message, Message List}
\]

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

**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:

$A \rightarrow B$: $N_A$, $A$

$B \rightarrow A$: $\{N_A, N_B, A\}_{K_{AB}}$

$A \rightarrow B$: $N_B$

$A \rightarrow B$: $N_A$, $A$

$B \rightarrow A$: $\{N_A, N_B, B\}_{K_{AB}}$

$A \rightarrow B$: $N_B$

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

$A \rightarrow B$: $N_A$, $A$

$B \rightarrow A$: $\{N_A, N_B, B\}_{K_{AB}}$

$A \rightarrow B$: $\{N_A, N_B\}_{K_{AB}}$

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

$A \rightarrow B$: $\{N_A, A\}_{K_B}$

$B \rightarrow A$: $\{N_A, N_B, B\}_{K_A}$

$A \rightarrow B$: $N_B$
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


<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Dates</th>
<th>Event Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS San Francisco Winter 2019</td>
<td>San Francisco, CAUS</td>
<td>Dec 02, 2019 - Dec 07, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Frankfurt December 2019</td>
<td>Frankfurt, DE</td>
<td>Dec 09, 2019 - Dec 14, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Austin Winter 2020</td>
<td>Austin, TXUS</td>
<td>Jan 06, 2020 - Jan 11, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Miami 2020</td>
<td>Miami, FLUS</td>
<td>Jan 13, 2020 - Jan 18, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>Cyber Threat Intelligence Summit &amp; Training 2020</td>
<td>Arlington, VAUS</td>
<td>Jan 20, 2020 - Jan 27, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Amsterdam January 2020</td>
<td>Amsterdam, NL</td>
<td>Jan 20, 2020 - Jan 25, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Anaheim 2020</td>
<td>Anaheim, CAUS</td>
<td>Jan 20, 2020 - Jan 25, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>MGT521 Beta Two 2020</td>
<td>San Diego, CAUS</td>
<td>Jan 22, 2020 - Jan 23, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Las Vegas 2020</td>
<td>Las Vegas, NVUS</td>
<td>Jan 27, 2020 - Feb 01, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Vienna January 2020</td>
<td>Vienna, AT</td>
<td>Jan 27, 2020 - Feb 01, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Francisco East Bay 2020</td>
<td>Emeryville, CAUS</td>
<td>Jan 27, 2020 - Feb 01, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Security East 2020</td>
<td>New Orleans, LAUS</td>
<td>Feb 01, 2020 - Feb 08, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Northern VA - Fairfax 2020</td>
<td>Fairfax, VAUS</td>
<td>Feb 10, 2020 - Feb 15, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS New York City Winter 2020</td>
<td>New York City, NYUS</td>
<td>Feb 10, 2020 - Feb 15, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Dubai February 2020</td>
<td>Dubai, AE</td>
<td>Feb 15, 2020 - Feb 20, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Brussels February 2020</td>
<td>Brussels, BE</td>
<td>Feb 17, 2020 - Feb 22, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS San Diego 2020</td>
<td>San Diego, CAUS</td>
<td>Feb 17, 2020 - Feb 22, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Scottsdale 2020</td>
<td>Scottsdale, AZUS</td>
<td>Feb 17, 2020 - Feb 22, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>Open-Source Intelligence Summit &amp; Training 2020</td>
<td>Alexandria, VAUS</td>
<td>Feb 18, 2020 - Feb 24, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Training at RSA Conference 2020</td>
<td>San Francisco, CAUS</td>
<td>Feb 23, 2020 - Feb 24, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Zurich February 2020</td>
<td>Zurich, CH</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Secure India 2020</td>
<td>Bangalore, IN</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Jacksonville 2020</td>
<td>Jacksonville, FLUS</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Manchester February 2020</td>
<td>Manchester, GB</td>
<td>Feb 24, 2020 - Feb 29, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Nashville 2019</td>
<td>OnlineTNUS</td>
<td>Dec 02, 2019 - Dec 07, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>