Formal Security Protocol Analysis

By Phillip H. Griffin – ISSA Fellow, Raleigh Chapter

Help is on the way. Help in finding flaws in cryptographic protocols before the bad people do. Help in selecting the most secure alternative for fixing a defect. Help in deciding whether a proposed protocol amendment can actually strengthen or weaken security. Help in gaining the assurance provided by security proofs long before a protocol is implemented and deployed.

From the far reaches of current theoretical computer science research, from the depths of “strand space” (figure 1), the faint glimmer of practical results of protocol security proofs are beginning to emerge. These results are based on a complex process: security threat model analysis. This analysis relies on the use of automated tools to detect protocol flaws and to help standards developers perfect the security standards that underlie our information systems and risk management controls.

Improving security standards

Royal Holloway, University of London, UK, hosted the first in a series of Security Standardization Research (SSR) conferences in December 2014. The conference program covered a wide variety of cybersecurity topics, including biometric authentication, standardized security techniques, key management, network security, smart card (ICC) security, and proposals to enhance transparency and openness in the processes used by standards development organizations. Accepted papers presented by industry and academic experts from around the world ranged in scope from theory to practice.

Several papers presented at the conference revealed new flaws discovered in cryptographic protocol standards. In some of these papers, the flaws were detected using formal symbolic protocol analysis tools. Such tools have been under development for many years. However, their use had been confined for the most part to the analysis of either very simple protocols or theoretical protocols of academic interest. Recently, these tools have shown a more practical side. Their increasing maturity has proved them capable of detecting flaws in protocols defined in national, international, and industry standards and deployed in commercial products.

Not all of the protocol security flaws described at the SSR-2014 conference were discovered using these new automated techniques. Several significant findings were the result of more traditional methods. These findings included flaws in standardized and deployed protocols, as well as important innovations to improve and extend the capabilities of long-used cryptographic techniques. Based on these results, we can look forward to products that eliminate these flaws and offer improved security and privacy.

Improvements and flaws: Traditional methods

Some researchers discovered protocol improvements and flaws without the use of automated techniques. Michael Ward of MasterCard, UK, presented “Blinded Diffie-Hellman” on behalf of EMVCo. His presentation introduced a new key agreement protocol for secure, privacy-preserving information exchange between a contactless payment card and a merchant terminal. This new protocol will enhance the performance of future financial transaction systems and better protect cardholders from stalking.

New payment systems based on Blinded Diffie-Hellman will use Elliptic Curve Cryptography (ECC) to enhance their performance, rather than relying on RSA-based schemes. This protocol has the added benefit of the card not being required to support an ECC signature (e.g., Elliptic Curve Digital Signature) algorithm or its associated cryptographic hash function. The new protocol will help prevent tracking cardholders through the repeated use of their public key certificates.
has been estimated that the processing time for Blinded Diffie-Hellman was reduced by thirty percent over that required using a similar station-to-station protocol.

Feng Hao of Newcastle University, UK, described issues found in the standardized versions of the Simple Password Exponential Key Exchange (SPEKE) protocol in a conference paper presentation. The paper’s authors discovered security flaws using traditional research and analysis techniques that did not rely on formal analysis tools. Though he made no claims of flaws discovered in specific applications, his presentation on “The SPEKE Protocol Revisited” revealed that SPEKE was subject to both key-malleability and impersonation attacks. SPEKE is a popular password-authenticated key exchange (PAKE) protocol that has been widely used since the middle 1990s. Both TruePass products from Entrust and BlackBerry devices from Research in Motion are commercial deployments based on the SPEKE protocol.

My own presentation of a “Standardization Transparency” paper described flaws in the schema definition and message designs used for secure information exchange in several international standards. Defects were evident from simply reading the schema published in these standards. These flaws are present in the ISO/IEC 19785-4 Common Biometric Exchange Format Framework (CBEFF) – Security block format, the International Civil Aviation Organization (ICAO) standard for electronic passports, and the ISO/IEC 24761 Authentication Context for Biometrics specifications.

Figure 2 – Schema definition flaws (Source: IEEE 52.1, p. 188)

Figure 3 – Message design flaws (Source: IEEE 52.1, p. 189)

The presentations I chaired in the session on “Analysis with Formal Methods” provided me with the most benefit. These presentations described new tools and analysis results that would set the tone of the conference. In addition, the panel discussion on “Formal Verification and Analysis of Protocols in Standards Development and Evolution” that followed provided a glimpse into the future of perfecting security protocols and cryptographic techniques.

Protocol analysis tools and techniques

The presentations I chaired in the session on “Analysis with Formal Methods” provided me with the most benefit. These presentations described new tools and analysis results that would set the tone of the conference. In addition, the panel discussion on “Formal Verification and Analysis of Protocols in Standards Development and Evolution” that followed provided a glimpse into the future of perfecting security protocols and cryptographic techniques.


don’t Miss This Web Conference!
Continuous Forensic Analytics: Issues and Answers
2-Hour Live Event
Scheduled for 9:00 am PDT, 12:00 pm EDT, 5:00 pm London, Tuesday, April 14, 2015.

Big Data, what good is it? Given all the data we have seen, it has become obvious that we can use it for more than we are currently thinking of. It can be used for predictive security analytics as well as for forensic analysis. With big data these can even be combined to be continuous forensic analytics. Please join us as we hear from Dipto Chakravarty, Geoff Hancock, and Tyrone Wilson.

Generously sponsored by

For more information or to register:

www.issa.org/?April2015T
Join the conversation: #ISSAWebConf

For more information on our webinar schedule:

From the 1980s well into the 1990s, formal security protocol analysis focused on Burrows–Abadi–Needham (BAN) logic. During this period, researchers developed threat models and performed security protocol analysis by hand. This approach was successful in discovering flaws and hidden assumptions in security protocols, the rules that govern information exchange between communicating parties that require origin authenticity, message integrity, and data confidentiality. This early success stimulated interest in the development of threat model checkers and theorem provers supported by formal protocol analysis tools. Several SSR-2014 conference papers presented results obtained from the application of some of these new tools.

**Cryptographic Protocol Shapes Analyzer**

The Cryptographic Protocol Shapes Analyzer (CPSA) is a software tool that attempts to create a list of all possible "shapes" of a cryptographic protocol. Each shape represents one possible protocol execution. The authentication and confidentiality properties of that protocol, as well as any anomalies and possible attacks, can be determined from the list of generated CPSA shapes. CPSA tools developed by the MITRE Corporation5 are freely available for non-commercial use (see sidebar).

CPSA supported the expression of protocol security goals in a paper describing an "enrich-by-need" analysis technique presented by Paul D. Rowe, "Security Goals and Evolving Standards" [2]. The paper proposed a minimalist, logical language of security protocol goals, and demonstrated its effectiveness by example, the mitigation of a known flaw discovered early in the evolution of the Kerberos PKINIT protocol. The authors noted that in order to achieve a decidable set of security goals, it was necessary to limit the expressiveness of the language. As such, their proposed language contained no data types, arithmetic expressions, or ability to define messaging syntax or schema required to implement and test actual security products.

**Maude NPA**

Maude is a powerful programming language used for modeling systems. Maude programs can model system operations, as well as how these operations interact. Category theory, a "general mathematical theory of structures and of systems of structures" [3], forms the basis of Maude semantics. In mathematics, category theory is an alternative to set theory. Categories are algebraic structures that when represented as functional definitions are useful in constructing security proofs of protocols.

Maude-NRL (Naval Research Laboratory) Protocol Analyzer (Maude-NPA) is a tool used to provide security proofs of cryptographic protocols and to search for protocol flaws and cryptosystem attacks. At the SSR-2014 conference, Catherine Meadows of NRL presented the results of new work that expanded the use of Maude-NPA to analyze the security ap-

---

5 http://hackage.haskell.org/package/cpsa

**MITRE CPSA**

The free MITRE CPSA tool distribution provides a tutorial example aimed at beginning users, the two step Blanchet key establishment protocol described in figure 4. This protocol has two roles, an initiator and a responder. Alice initiates the protocol by creating a fresh symmetric key S. She signs S with the private component of her public-private asymmetric key pair. Alice then encrypts the signed key with the public component of Bob’s public-private key pair, $K_B$. Alice sends the signed and encrypted key in a message to Bob.

$$A \rightarrow B : \{ \{ S \}^{x} \}_{K_B}$$

$$B \rightarrow A : \{ D \}^{y}_{S}$$

**Figure 4– Alice sends Bob a signed and encrypted key, and Bob responds with encrypted data**

Bob receives the encrypted message from Alice. He decrypts the cipher text using the private key component of his public-private key pair to recover the signed symmetric key S. Bob then validates the signature on S using the public component of Alice’s public-private key pair to gain assurance that S has not been tampered with and that it was signed by Alice. Bob now encrypts some data D using S and sends the encrypted data to Alice, who can recover the plain text data using S, the same symmetric key she established with Bob.

The Blanchet protocol represented in CPSA in figure 5 is a set of roles and a problem description of what we assume happens when we operate the protocol described in figure 4. We can create the following CPSA example with a simple text editor:

```
(defprotocol blanchet basic
  (defrole initiator
    (vars (a b name) (s skey) (d data))
    (trace
     (send (enc (enc s (privk a)) (pubk b)))
     (recv (enc d s)))
    )
  )
  (defrole responder
    (vars (a b name) (s skey) (d data))
    (trace
     (recv (enc (enc s (privk a)) (pubk b)))
     (send (enc d s))
    )
  )
)
```

**Figure 5 –Blanchet protocol in CPSA**

(Source: MITRE Haskell CPSA package)

On a Windows platform, the user then clicks an icon that executes a `Make.hs` command on the resulting text file and types “build” to analyze the Blanchet protocol. The user can view the CPSA output in a web browser to examine what the tool inferred actually happened when the protocol was operated, and what steps CPSA took to produce these results. From there, the tutorial takes the user through a detailed explanation of the CPSA results, then step by step though further protocol refinements, modeling, and security analysis.
application-programming interface (API) of a hardware security module (HSM), the IBM 4758. The research examined the Common Cryptographic Architecture (CCA) API of this HSM device. Her presentation described a paper, “Analysis of the IBM CCA Security API Protocols in Maude-NPA” [2], in which the authors were first ever to apply a general-purpose tool to XOR-based API analysis, rather than developing special purpose solutions.

**Tamarin and Scyther**

The Tamarin prover\(^6\) is an open-source tool for security protocol analysis and verification. The functional programming language Haskell\(^7\) implements Tamarin, using a backend environment based on the Maude System\(^8\) and language. Tamarin relies on a symbolic message theory model of Diffie-Hellman exponentiation and supports unbounded verification or falsification of security protocols. It can both find new attacks automatically and verify the correctness of protocols with respect to a model.

Tamarin is in the same family of tools as Scyther, another freely available protocol analysis tool. The application of a tool set combining Tamarin and Scyther found flaws in the ISO/IEC 9798-2 Entity Authentication standard. These findings led to the release of an updated version of the standard to correct defects identified in one version of a mutual authentication protocol. Unfortunately, these findings were not used at that time to detect similar faults in ISO/IEC 11770 key establishment protocols, even though these are derived from ISO/IEC 9798 authentication protocols.

Improvements to ISO/IEC 11770 would have to wait for future work by Cas Cremers and Marko Horvat of the University of Oxford, presented in a paper\(^9\) at the SSR-2014 conference. In “Improving the ISO/IEC 11770 Standard for Key Management Techniques” [2] the authors made use of Scyther to uncover incorrect security properties (i.e., protocol security claims) and to inform suggestions for future security improvements. The use of automated techniques allowed them to examine variants of some 30 protocols in the 2014 edition of the standard. Their security protocol models and Scyther tools are freely available for download.\(^10\)

**Standards defect process problems**

The scope of what researchers are looking for can limit results obtained from formal protocol analysis models. A model constructed to detect security flaws may be blind to defects that limit privacy protections. Analysts may not examine privacy features, even when the protocol being examined claims to offer privacy protections. This was the case with the Protocol for Lightweight Authentication of Identity (PLAID), which is currently being fast-tracked as international standard ISO/IEC 9798-2.

---

7 [https://wiki.haskell.org/Haskell](https://wiki.haskell.org/Haskell)
10 [http://www.cs.ox.ac.uk/people/cas.cremers/scyther/](http://www.cs.ox.ac.uk/people/cas.cremers/scyther/)
IEC DIS 25182-1. PLAID intends to provide a secure and private authentication protocol between a smart card (ICC) and terminal.

The PLAID protocol attempts to find the middle ground between fast tag-based RFID solutions that transfer data in the clear and slower PKI-based solutions that protect data and provide authentication at the cost of speed. PLAID can support multi-factor authentication solutions that pair standards-based cryptography commonly available on ICCs with biometric technologies and cardholder PINs. Security aspects of PLAID were examined to some extent using the ProVerif\textsuperscript{12} and Scyther modeling tools, but this research did not include analysis of PLAID’s privacy claims.

PLAID is a national standard being promoted by the Australian Department of Defense and Department of Human Services. These agencies promote PLAID adoption by providing unencumbered intellectual property and a freely available reference implementation.\textsuperscript{13} In the US, NIST has promoted the adoption of PLAID.\textsuperscript{14} The US also heads the ISO/IEC group that manages the fast-track processing of the draft PLAID standard. Responsibility for developing the ISO/IEC DIS 25182-1 standard has not been assigned to the SC 17 Cards and Personal Identification subcommittee that usually develops such card security standards.

As reported at the SSR-2014 conference in the presentation by Victoria Fehr on "Unpicking PLAID,"\textsuperscript{15} “the privacy properties of PLAID are significantly weaker than claimed” [2]. Using standard statistical and data cryptanalysis techniques, security researchers were able to fingerprint, and then later identify cards. The attack allowed an adversary to determine all of the cryptographic keys supported by the smart card. These security and privacy defects were reported to the standards development committee responsible for PLAID, along with proposed corrections. However, this action has not resulted in any corrective changes being made to the draft PLAID standard. Based on changes made in subsequent revisions, it appears that the standard will be progressed without addressing the reported defects.

The benefits of research findings may not always find their way into security protocol standards, even when they have been presented to the committees of standards development organizations. In another event, analysis results found in the ISO/IEC 9798 Entity Authentication standard using the Scyther tool in 2012 were reported, and several problems were identified and corrected. However, these results could have been applied years earlier to the ISO/IEC 11770 Key Management standard to protocols that are derived from those in ISO/IEC 9798.

Future work

How can formal protocol analysis benefit information security practitioners today? Though formal analysis tools are improving rapidly, they are not yet able to be of direct benefit in helping us to secure our systems. Their primary benefit today lies in their ability to detect flaws in protocols defined in information security standards, and in assisting standards developers in choosing the most secure fixes from a set of alternatives when correcting defects.

The next Security Standardization Research 2015\textsuperscript{16} conference this year in Tokyo, Japan, may yield more progress. However, formal protocol analysis is not yet a requirement of standards bodies, and security proofs are not likely to be included in standards texts. As current research evolves and

Formal Security Protocol Analysis | Phillip H. Griffin

As security researcher Alfred Menezes wisely cautions, any claim that “a cryptographic protocol is ‘provably secure’ should always be met with a healthy dose of skepticism” [4]. As the case of “Unpicking PLAID” revealed, we should ensure that the protocol model used to provide assurance covers all, and not just some, of its claims. We must question and thoroughly test the validity of all assumptions that support the model. The proof should cover the types of attacks and “attackers that one could expect in the environments in which the protocol will eventually be deployed” [4]. Finally, we need to ensure that the correctness of the proofs we rely on are checked and their claims of correctness adequately verified.

We should also carefully examine the design and validity of the security protocol messages we rely on. Standards development organizations (SDOs) should correct cryptographic message design defects such as those illustrated in “Security Standardization Transparency,” and implementations based on these defective standards should be repaired. SDOs should review continued reliance on core security standards whose schema validity defects have remained incorrect since the 1990s. Reference to suitable substitutes should replace reference to any flawed standards where possible.

We should standardize definitions and notation used to construct security protocol models, so that models used for one protocol can be compared with others. We should make it possible to more easily modify and apply models to similar protocols. SDOs should be encouraged to include security proofs based on standardized models in their work, and these models should guide the design, development, and testing of vendor products. Standardization of notation and models may allow security protocol analysis models to one day feed into protocol test tools practitioners need for gaining assurance that vendor products implement these protocols correctly.

References


About the Author

Phillip H. Griffin, CISM, has over 20 years experience in the development of commercial, national, and international security standards and cryptographic messaging protocols. Phil has a Master of Information Technology, Information Assurance and Security degree and he has been awarded nine US patents at the intersection of biometrics, radio frequency identification (RFID), and information security management. He may be reached at phil@philipgriffin.com.

Figure 6 – Two notations for encrypting message M with key K

A standardized model representation could have many benefits. Having standardized models might encourage standards bodies to include them in their publications. A standard modeling notation would allow comparison of models and make them easier to modify and apply to similar protocols. In addition, a standard model notation could make it easier to combine current and future tools for protocol analysis. The current tool-specific models could be mapped to a standard model without requiring tool modifications.

On the other end of the process, where implementers build products that security professionals use to mitigate security risk, there are other gaps to bridge. Test case messaging needs to flow from protocol analysis tools. Such messages defined in a standardized schema can provide product implementers with test cases and expected test case results for each protocol examined. By successfully passing or failing test cases as expected by formal protocol analysis, product developers can gain assurance that they have correctly implemented a given security protocol. Standardized protocol test case suites might pave the way for product validation and certification activities, such as those offered by NIST and other organizations.

Conclusions

As security researcher Alfred Menezes wisely cautions, any claim that “a cryptographic protocol is ‘provably secure’ should always be met with a healthy dose of skepticism” [4]. As the case of “Unpicking PLAID” revealed, we should ensure that the protocol model used to provide assurance covers all, and not just some, of its claims. We must question and thoroughly test the validity of all assumptions that support the model. The proof should cover the types of attacks and “attackers that one could expect in the environments in which the protocol will eventually be deployed” [4]. Finally, we need to ensure that the correctness of the proofs we rely on are checked and their claims of correctness adequately verified.

©2015 ISSA • www.issa.org • editor@issa.org • Permission for author use only.