Improving the Security of Cryptographic Protocol Standards

David Basin Cas Cremers Kunihiko Miyazaki
Sasa Radomirovic Dai Watanabe

May 1, 2014

Keywords: Standards, Network Protocols, Security, Protocol Verification, Formal Methods

1 Security Protocol Standards

Security protocols are distributed algorithms that use cryptography to achieve security objectives. In practice, these protocols regulate how computing devices carry out security-critical tasks. For example, Transport Layer Security (TLS) is used to establish secure communication channels between clients and servers, Kerberos is used for distributed authentication and authorization, and Internet Protocol Security (IPsec) can be used to set up virtual private networks. These protocols are omnipresent and are used to access and protect numerous applications, ranging from banking to social media. There are also many lesser known protocols in use, such as WiMAX for secure communication in wireless networks, ISO/IEC 9798 for entity authentication, and EAP for network access authentication.

When a protocol like TLS is used, any client should potentially be able to communicate with any server, independent of the operating system they run on or the programming language used for their implementation. This generality is enabled by standards and technical documents such as Request for Comments (RFCs). These documents describe the protocol's operation in sufficient detail to guide the construction of interoperable implementations. All of the protocols mentioned above are described by standards or RFCs, which have been approved by standardization bodies or are undergoing standardization.

A closer look at modern protocol standards indicates that while standardization bodies are doing excellent work, the security of the resulting protocols varies considerably. Over the past decade, we have carried out numerous case studies with model-checking tools for security protocols, some of which we have developed ourselves [4, 6, 14, 16]. Our analysis shows that many standards suffer from security weakness including basic mistakes and well-known flaws. In some cases, the weaknesses have been quite serious. But even minor problems are best avoided from the start, prior to standardization. It is time-consuming to amend standards. And after amendment, companies with products implementing the standard must decide between costly product upgrades or running the risk of loss of reputation and litigation for distributing products with known defects.

Because standards are carefully designed by experts, one might expect them to meet strong, well-understood, and well-specified security guarantees. Unfortunately, this expectation is not always met. While standards often contain detailed functional descriptions, many standards do not contain much information about security guarantees. Instead of unambiguous security properties and clear threat models, many cryptographic protocol standards specify, at best, high-level security properties and a handful of threat scenarios. This lack of clear threat models and specified properties makes it impossible to objectively assess a protocol's merits: without them, there is nothing to objectively verify or falsify.

During the last few decades, formal methods have been successfully used to analyse small, academic protocols with well-defined threat models (also called adversary models) and clear security goals. More recently, researchers from the formal methods community have analyzed several protocol standards. This process has typically involved proposing threat models and security properties, and analyzing the standard with respect to the properties conjectured by the researchers.

In what follows, we illustrate the kinds of problems that arise when security properties and threat models are neglected in standards, and demonstrate how formal methods can make a difference by presenting several case studies. Afterwards, we step back to see how formal methods and associated tools could be better integrated in the standardization process and what the present obstacles and limitations are. Our case studies are based on the three previously mentioned protocols, WiMAX, EAP, and ISO/IEC 9798.

2 WiMAX

Our first case study is the wireless communication standard, IEEE 802.16, also known as WiMAX, which aims to enable the delivery of "last mile" wireless broadband access. The WiMAX standard includes several mechanisms that deal with keys or involve cryptographic operations. The core mechanism is the authorization phase, which establishes a shared secret, on which all subsequent security is based. This authorization can be performed using EAP protocols, or alternatively, using the PKM protocols described by the standard.

The WiMAX standard was originally proposed in 2001 and has been updated several times since then. The first version includes only the PKMv1-RSA protocol. This protocol is executed between a subscriber station (SS), typically an end user's WiMAX modem, and a service provider's base station (BS). At a high level, the protocol proceeds as follows. The subscriber station initiates communication with the base station by sending its certificate, the list of algorithms that it supports, and a unique connection identifier (CID). The base station generates an authorization key (AK) and sends this back encrypted with the subscriber station's public key. It also sends the key's sequence number and lifetime, and a security association identifier, which we will collectively denote by SAID in the following. These message exchanges can be represented as follows.

(PKMv1-RSA)

$$\begin{array}{ll} \text{SS} \rightarrow \text{BS}: & \text{SS\_Certificate, SS\_Algo\_Suites, CID} \\ \text{BS} \rightarrow \text{SS}: & \text{Enc}_{\text{PK(SS)}}(\text{AK}), \text{SAID} \end{array}$$

After the standard's initial release, Johnston and Walker in 2004 [9] identified several weaknesses. In particular, they argued that PKMv1-RSA essentially provides no security guarantees since, in the context of wireless transmissions, one should assume that attackers can *spoof* (i.e. send messages impersonating another party) arbitrary messages. The subscriber station therefore has no idea who encrypted, or even generated, the key it receives.

Johnston and Walker argued the protocol should at least provide mutual authentication under the (realistic) assumption that attackers can eavesdrop and inject wireless network traffic. Their arguments were necessarily informal since the standard specifies neither a threat model nor any details about the security properties it aims to achieve. Furthermore, whereas Johnston and Walker were specific about the weaknesses in PKMv1-RSA, "mutual authentication" is not a uniquely defined concept either, as there are many possible definitions of authentication of varying strengths, as illustrated in the sidebar.

In 2005, a new version of the standard was released that introduced the PKMv2-RSA protocol. The new version is a three-message protocol where all messages are digitally signed. The subscriber station initiates communication with the base station by sending a random number (SS_Random), its certificate, and a unique connection identifier. The message is signed with the subscriber station's private RSA key (SigSS). The base station generates a key (pre-PAK), concatenates it with the subscriber station's MAC address (SS_MAC), and encrypts the result with the subscriber station's public key. It sends back to the subscriber station this encrypted message together with the subscriber station's random number, its own random number, and its certificate. The message is signed with the base station's private key (SigBS). In the third message, the subscriber station confirms the receipt of the previous message by sending back the base station's random number and signing the message (SigSS').

(PKMv2-RSA)

$$\begin{array}{ll} \text{SS} \rightarrow \text{BS}: & \text{SS\_Random, SS\_Certificate, CID, SigSS} \\ \text{BS} \rightarrow \text{SS}: & \text{SS\_Random}, \text{Enc}_{\text{PK(SS)}}(\text{pre-PAK}||\text{SS\_MAC}), \\ & \text{BS\_Random, SAID, BS\_Certificate, SigBS} \\ \text{SS} \rightarrow \text{BS}: & \text{BS\_Random, SigSS'} \end{array}$$

It appears that this new protocol was intended to address the weaknesses in PKMv1-RSA. But again, neither a threat model nor security properties were specified in the standard. Consequently, even though the numbering may

The ambiguity of authentication (SIDEBAR)

Authentication is a very common security goal. However, this notion has numerous substantially different interpretations, each with several variants. Below are three typical interpretations of *a client C is authenticated by a server S*, each with a weaker and a stronger variant.

Entity Authentication

- Aliveness of C: C has performed an action.
- Recent aliveness of C: C has performed an action (causally) after a specific action of S.

Data Agreement

- Non-injective agreement on message m: S has received the message m from C. C has sent m to S.
- Agreement on message m: Non-injective agreement on m and S will not accept m if it is replayed by the adversary.

Authenticated Session Key

- Authenticated session key k: Session key k is a fresh session key, known only to C and S and possibly some trusted third party.
- Authenticated session key k with compromise resilience: k is an authenticated session key and compromise of an old session key does not lead to compromise of k.

For each of these interpretations, many more variants exist. The critical observation is that there is no one "right" definition of authentication: one cannot specify an appropriate authentication property without a fundamental understanding of the application scenario.

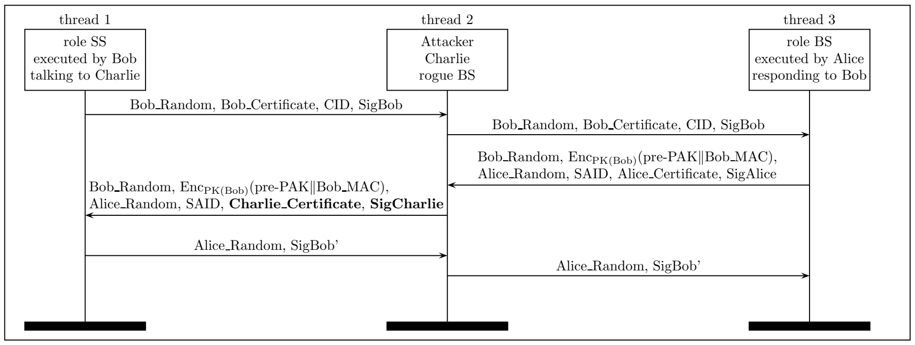
thread 1
role SS executed by Bob talking to Charlie
thread 2
Attacker Charlie rogue BS
thread 3
role BS executed by Alice responding to Bob
Bob_Random, Bob_Certificate, CID, SigBob
Bob_Random, Bob_Certificate, CID, SigBob
Bob_Random, Enc$_{PK(Bob)}$(pre-PAK‖Bob_MAC), Alice_Random, SAID, Alice_Certificate, SigAlice
Bob_Random, Enc$_{PK(Bob)}$(pre-PAK‖Bob_MAC), Alice_Random, SAID, **Charlie_Certificate**, **SigCharlie**
Alice_Random, SigBob'
Alice_Random, SigBob'

Figure 1: Man-in-the-middle attack on PKMv2-RSA. Bob is talking to Charlie. Alice thinks Bob is talking to her.

suggest that PKMv2-RSA provides additional properties to those provided by PKMv1-RSA, the standard provides no concrete statements to this effect.

Both academic and industrial experts were involved in a security review of drafts of the 2005 version of the standard [1] using manual inspection. These reviews led to changes that found their way into the revised standard. However, soon after the new version's release, researchers pointed out that an "interleaving attack" exists on the PKMv2-RSA protocol [17]. This is a common-place man-in-the-middle attack where the attacker forwards and selectively modifies messages between two parties. In 2008, the formal protocol analysis tool Scyther was used to analyze several subprotocols from the standard [2]. The man-in-the-middle attack from [17] was independently rediscovered, a fix was proposed, and its correctness verified.

The attack is shown in Figure 1 and proceeds as follows. The adversary controls a rogue base station which we will call *Charlie*. When a subscriber *Bob* tries to establish a connection with Charlie, the adversary reroutes the message to the legitimate base station *Alice* instead. Alice replies with a cryptographically signed message, thinking that Bob is trying to start a session with her. Her message contains an encrypted key for Bob. The adversary re-signs Alice's reply with Charlie's private key and sends it on to Bob. Bob responds as expected, and the adversary reroutes the message again to Alice. In the end, Alice correctly thinks that she is communicating with Bob, but Bob thinks he is talking to Charlie. Thus, authentication of the session's participants fails.

The cause of this problem is that the first and third messages do not include any information on the subscriber's assumptions on who the base station is. The attack can be prevented simply by adding the base station's identity to the third message, as proposed in [2].

Interestingly, despite this attack, the adversary can neither eavesdrop on Bob's subsequent messages nor send messages impersonating Bob. The reason is twofold. First, the adversary cannot decrypt the key Alice sends to Bob. Second, the protocol immediately following PKMv2-RSA cryptographically binds the communication partners' identities to all exchanged messages. Thus the adversary cannot continue his attack. So is this "attack" really a security threat? The surprisingly simple answer is that, because the standard neither specifies the intended security properties nor the threat model, we cannot know for sure. If we play it safe assuming that PKMv2-RSA does not provide strong security guarantees, then the cryptographic operations performed in PKMv2-RSA and the subsequent protocol are simply redundant overhead. In fact, we can discard the third message of PKMv2-RSA without sacrificing the security properties that it achieves in composition with the subsequent protocol [2]. We could therefore simplify the protocol and reduce its communication complexity. Alternatively, we can accept that PKMv2-RSA is intended to be a three-message authentication protocol and ignore the man-in-the-middle problem. This could, however, lead to real problems, should PKMv2-RSA be combined with a different subsequent protocol where the protocol engineers rely on the statement in IEEE 802.15e that PKMv2-RSA achieves mutual authentication.

Unfortunately, this lack of specification of the threat model and security properties is not an isolated case.

3 EAP

Our second case study is the Extensible Authentication Protocol (EAP) developed by the Internet Engineering Task Force (IETF). Compared to other standardisation bodies, the IETF uses a completely public process for developing standards. There is no formal membership, the standardization process is open to all parties, and its publications, including the RFCs and Internet drafts referred to in the following, are freely available online (`http://www.ietf.org`). This enables us to study the evolution of EAP, which is currently an IETF *proposed standard*.

EAP is a framework for network access authentication. It supports multiple authentication protocols, known as *methods*. Some of the better known EAP authentication methods are EAP-TLS, EAP-SIM, and EAP-AKA, used for authentication and session key distribution in WPA/WPA2, GSM, and UMTS networks, respectively.

EAP started out in 1995 as an Internet draft for the PPP Extensible Authentication Protocol. PPP is the point-to-point protocol, first published as an RFC in 1989. In April 2004, an Internet draft document was published reviewing 48 EAP authentication methods. Some of these methods were found not to be under active development and many were found not to comply with the then evolving EAP reference document, which became RFC 3748. Of the remaining methods, several interesting candidates were identified, but their comparison was left for future work. A comparison then would have been difficult, since an EAP threat model and specific security claims were only introduced in RFC 3748. In fact, even given the threat model and security claims of RFC 3748, we still consider it a challenge to compare EAP authentication methods because the threat model is too vague.

The threat model is defined by the assumption that an attacker may compromise links over which EAP packets are transmitted and by a list of ten attacks. This is, of course, a source of ambiguity: any attack that is not specifically mentioned could be considered out of the threat model's scope. Examining the ten attacks closer, one sees that they mix generic attacker capabilities with specific attack scenarios. For instance, the first states that the attacker can eavesdrop on the communication link, but narrows this ability down to discovering user identities. The second affords the attacker two generic capabilities, namely spoofing and modification of packets, but restricted to EAP packets.

One way to obtain a more precise threat model is to focus on what we consider to be the essential attacker capabilities. From the first two items on the list we infer: An attacker can eavesdrop on, spoof, and modify EAP packets. Several of the subsequent items on the list consider specific attack scenarios that are consequences of these three capabilities: One concerns denial of service attacks by spoofing messages and three others concern specific man-in-the-middle attacks. The last item considers a particular scenario where an attacker may spoof lower-layer protocol messages. The attacker's capability in this case is not defined by the particular scenario, but by the fact that lower layer messages are also considered to be under the attacker's control. Thus, we can infer that an attacker is assumed to be able to eavesdrop on, spoof, and modify EAP and all lower layer packets. The remaining items state that an attacker can perform offline computations, such as dictionary attacks on passwords and attacks on weak cryptographic schemes.

We now turn to the security properties. An EAP authentication method specification must state which security properties it claims to satisfy by referring to a non-exhaustive list given in Section 7.2.1 of RFC 3748. It is recommended that the claims are supported with evidence in the form of a proof or a reference. Below are a selection of properties relevant for making precise statements about a protocols' behavior. The properties' descriptions are lightly edited quotes from Section 7.2.1 of RFC 3748.

Integrity protection. This refers to data origin authentication and protection against unauthorized modification of information for EAP packets (including EAP Requests and Responses). When making this claim, a method specification must describe the EAP packets and their fields that are protected.

Replay protection. This refers to protection against replay of an EAP method or its messages, including status messages.

Session independence. The demonstration that passive attacks (such as capture of the EAP conversation) or active attacks (including compromise of the master session keys) do not enable compromise of subsequent or prior keys.

Even though no clear threat model is given, these descriptions match well with established concepts from the verification community. Integrity protection is related to *data agreement*, replay protection is related to *injectivity*, and session independence to *backward* and *forward secrecy*.

The confidentiality claim, surprisingly, is based on a definition that unnecessarily complicates the analysis and comparison of protocols:

> **Confidentiality.** This refers to encryption of EAP messages, including EAP Requests and Responses, and status indications. A method making this claim must support identity protection (see RFC 3748, Section 7.3).

There are two problems with this property. First, in an adversarially controlled network, encryption is necessary to ensure the confidentiality of messages, but it is not sufficient in general. An artificial, but striking example was constructed by Dolev and Yao [7]. It demonstrates how a secure communication protocol, employing public-key cryptography, can be turned into an insecure protocol simply by encrypting every protocol message an additional time.

Second, to satisfy this property, an authentication method must not only provide message confidentiality, but also "identity protection", a privacy feature that arguably is an unrelated property. The consequence of having these two distinct properties combined into one is that authentication methods which provide confidentiality of messages, but not identity protection, e.g. EAP-PSK (RFC 4764), cannot be easily distinguished from authentication methods, e.g. EAP-MD5-Challenge (RFC 3748), that provide neither of the two properties.

RFC 3748 has been updated by RFC 5247 in which the threat model is clearer, but RFC 5247 does not update the security claims. Still, there is a clear development towards a more precise security model. Moreover, an IETF *best current practices* document published in 2007 (RFC 4962) advocates the use of formal methods in addition to expert review in the standardization process of key management protocols. In the next section, we illustrate the feasibility and benefits of employing formal verification methods in the context of a cryptographic protocol standard.

4 ISO/IEC 9798

Our final case study is the ISO/IEC 9798 standard for entity authentication protocols. The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) jointly develop standards for Information Technology. In 1991 they published the first part of the 9798 standard, which specifies a family of entity authentication protocols. This standard is mandated by numerous other standards that require entity authentication as a building block. Examples include the Guidelines on Algorithms Usage and Key Management by the European Committee for Banking Standards and the ITU-T multimedia standard H.235.

Since 1991, parts of the standard have been revised several times to address weaknesses and ambiguities. One may therefore expect that such a mature and pervasive standard is "bullet-proof" and that the protocols satisfy strong, practically relevant, authentication properties.

$$\begin{array}{lll} 1. & A \rightarrow B: & TN_A \parallel Text_2 \parallel f_{K_{AB}}(TN_A \parallel I_B \parallel Text_1) \\ 2. & B \rightarrow A: & TN_B \parallel Text_4 \parallel f_{K_{AB}}(TN_B \parallel I_A \parallel Text_3) \end{array}$$

Figure 2: The 2009 version of the ISO/IEC 9798 two-pass mutual authentication protocol using a cryptographic check function.

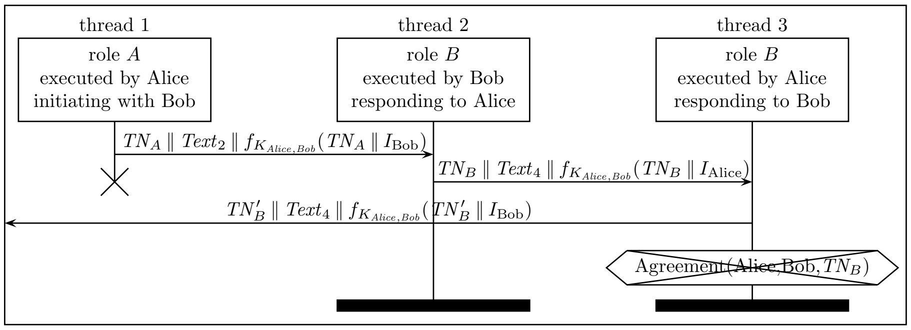
thread 1
thread 2
thread 3
role A executed by Alice initiating with Bob
role B executed by Bob responding to Alice
role B executed by Alice responding to Bob
$TN_A \parallel Text_2 \parallel f_{K_{Alice,Bob}}(TN_A \parallel I_{\text{Bob}})$
$TN_B \parallel Text_4 \parallel f_{K_{Alice,Bob}}(TN_B \parallel I_{\text{Alice}})$
$TN'_B \parallel Text_4 \parallel f_{K_{Alice,Bob}}(TN'_B \parallel I_{\text{Bob}})$
Agreement(Alice,Bob,TN_B)

Figure 3: Role-mixup attack on the 2009 version of the two-pass mutual authentication protocol using a cryptographic check function: when Alice finishes thread 3, she wrongly assumes that Bob was performing the A role.

However, it is not entirely clear which security properties are provided by the protocols in the standard. The standard claims to provide "entity authentication", alternatively phrased as "authentication of the claimant identity". As explained in the sidebar, there are several possible interpretations of these notions and this makes it extremely difficult for users of the standard to judge if a particular protocol provides a sufficient form of authentication. Similarly, as is common in many standards, the threat model is only defined in terms of specific (informal) attack types, such as "replay attack".

We became involved in an evaluation of this standard in 2010 by CRYPTREC, the Cryptography Research and Evaluation Committee set up by the Japanese Government. We formally analyzed the 2010 versions of the protocols specified in Parts 1–4 of the ISO/IEC 9798 standard using the Scyther tool [6]. For the threat model, we used the established "Dolev-Yao" model, in which the attacker has full control over the network, but cannot break the cryptographic primitives. We evaluated the protocols with respect to a subset of the authentication properties defined in [10].

To our surprise, we found that several weaknesses that had been previously reported in academic literature were still present in the standard. Moreover, we found new weaknesses. We provide one illustrative attack, called a *role-mixup attack*. In this attack, an agent's assumptions on another agent's role are wrong. The two data agreement properties (see sidebar) require that when Alice finishes her role apparently with Bob, then Alice and Bob not only agree on the exchanged data, but additionally Alice can be sure that Bob was performing in the intended role. Role-mixup attacks violate agreement properties.

Figure 3 shows an example of a role-mixup attack on the protocol from Figure 2. Agents perform actions such as sending and receiving messages, resulting in message transmissions represented by horizontal arrows. Actions are executed within threads, represented by vertical lines. The box at the top of each thread denotes the parameters involved in the thread's creation. Claims of security properties are denoted by hexagons and a crossed-out hexagon denotes that the claimed property is violated.

In this attack, the adversary uses a message from Bob in role B (thread 2) to trick Alice in role B (thread 3) into thinking that Bob is executing role A and is trying to initiate a session with her. However, Bob (thread 2) is replying to a message from Alice in role A (thread 1), and is executing role B. The adversary thereby tricks Alice into thinking that Bob is in a different state than he actually is.

Additionally, when the optional text fields $Text_1$ and $Text_3$ are used, the role-mixup attack also violates the agreement property with respect to these fields: Alice will end the protocol believing that the optional field data she receives from Bob was intended as $Text_1$, whereas Bob actually sent this data in the $Text_3$ field. Depending on the use of these fields, this may be a serious security problem. For example, consider a deployment scenario in which the optional text fields represent numbers. Let the first message be used for a transaction request where $Text_1$ represents the amount of money to be transferred. Assume the second message is used for confirmation where $Text_3$ corresponds to the transaction number. In this case, the adversary can re-use a response message, which contains a transaction number N, to insert a seemingly valid transaction request for the amount N.

Note that exploiting these attacks, as well as the other attacks we found, does not require "breaking" cryptography. Rather, the adversary exploits similarities among messages and the willingness of agents to engage in the protocol.

We analyzed the shortcomings in the protocols' design and proposed repairs, which we formally verified [3]. Our repairs address all the known problems. Based on our analysis, the ISO/IEC working group responsible for the 9798 standard released an updated version of the standard, incorporating our proposed protocol fixes in 2012.

We believe that the approach we have taken to analyze and provably repair the ISO/IEC 9798 standard can play an important role in future standardization efforts. Our approach supports standardization committees with both falsification, that is, finding errors in the early phases of standardization, and verification, providing objective and verifiable security guarantees in the end phases.

5 Discussion and Recommendations

Our three case studies suggest a trend towards an improved standardization process. The WiMAX study provides a cautionary tale on what happens when threat models and security goals are not included in the standard. In this case, the lack of these models created a situation where some protocols could be declared neither secure nor insecure and simple security flaws were not caught until late in the standardization process, requiring time-consuming, expensive amendments. The EAP case study indicates that security protocols are increasingly considered in the context of a threat model and are designed to satisfy specific security claims. However, the threat models and security claims tend to be specified informally, making it hard to compare protocol proposals and to decide whether a protocol is suitable for a given purpose. The ISO/IEC 9798 case study demonstrates that it is feasible to provide systematic threat models and precise security properties, and to perform formal verification. It also shows that formal methods are slowly starting to have an impact in standardization bodies, e.g., [3, 12, 13]. We expect this trend to continue as governments and other organizations increasingly push for the use of formal methods for the development and evaluation of critical standards.

For example, ISO/IEC JTC 1/SC 27 started in 2007 the project "Verification of cryptographic protocols" which involves developing a standard (ISO/IEC 29128) for certifying cryptographic protocol designs, where the highest evaluation levels require the use of formal, machine checked correctness proofs [11]. The four cornerstones of the ISO/IEC 29128 [8] certification process are the requirements that a security protocol document must contain a protocol specification, a threat model or adversary model, security properties, and self-assessment evidence. The specifics for these requirements depend on what protocol assurance level (PAL) is sought. At the lowest assurance level, PAL1, informal descriptions may be given for the protocol specification, adversary model, and security properties. The self-assessment may be conducted with informal arguments (PAL1) or mathematical "paper and pencil" proofs (PAL2) demonstrating that the security properties hold with respect to the adversary model. The higher levels require formal descriptions, specific to the automated tools employed to obtain the self-assessment evidence.

Unsurprisingly, we think that security protocol designs that are documented to satisfy the requirements sketched above would make for much improved security protocol standards. The current security protocol standardization process is still far from ideal and it will not change overnight. There are several gaps to bridge before formal verification becomes a standard procedure. Some of these gaps are due to cultural and technical language differences between network engineers, cryptographers, and other security researchers. Other gaps are in our own backyard and concern the automated tools employed in the verification process. For wide-spread industrial use, these tools must be robust, well-documented, and go beyond the current research prototypes. Moreover, the tools themselves must ultimately be certified to be correct.

We believe there is much work to be done before engineers are as comfortable specifying security properties and threat models as they are specifying functional requirements. Across domains, both security properties and threat models tend to be formulated at different abstraction levels and from very different perspectives. Addressing this requires research leading to a common framework for specifying security properties and threat models. Ideally, there should be a standardized set of unambiguous security properties and threat models, which can be referred to by other standards.

Once standards add unambiguous security properties and threat models to their functional specifications, we will have the foundations for evaluating the security merits of the standard, and subsequently for comparing different proposals, possibly using tool support [5, 6, 15].

References

[1] B. Aboba. Summary of the IEEE 802.16e D8 security review, Sept. 2005. `http://www.ieee802.org/16/tge/contrib/C80216e-05_373.pdf`.

[2] S. Andova, C. Cremers, K. Gjøsteen, S. Mauw, S. Mjølsnes, and S. Radomirović. A framework for compositional verification of security protocols. *Information and Computation*, 206:425–459, February 2008. The protocol models used in the analysis are available from `https://github.com/cascremers/scyther/tree/master/gui/Protocols/IEEE-WIMAX`.

[3] D. Basin, C. Cremers, and S. Meier. Provably repairing the ISO/IEC 9798 standard for entity authentication. In P. Degano and J. D. Guttman, editors, *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*, volume 7215 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2012.

[4] D. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, June 2005. Published online December 2004.

[5] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. 14th IEEE Computer Security Foundations Workshop (CSFW)*, pages 82–96. IEEE, 2001.

[6] C. Cremers. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proc. CAV*, volume 5123 of *LNCS*, pages 414–418. Springer, 2008. Available for download at `http://people.inf.ethz.ch/cremersc/scyther/`.

[7] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[8] International Organization for Standardization, Genève, Switzerland. ISO/IEC 29128:2011 Information technology - Security techniques - Verification of cryptographic protocols, 2011.

[9] D. Johnston and J. Walker. Overview of IEEE 802.16 Security. *IEEE Security and Privacy*, 2(3):40–48, May 2004.

[10] G. Lowe. A hierarchy of authentication specifications. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 31–44. IEEE, 1997.

[11] S. Matsuo, K. Miyazaki, A. Otsuka, and D. A. Basin. How to evaluate the security of real-life cryptographic protocols? - the cases of ISO/IEC 29128 and CRYPTREC. In *Financial Cryptography and Data Security, FC 2010 Workshops, RLCPS, WECSR, and WLC 2010, Spain, January 25-28, 2010, Revised Selected Papers*, volume 6054 of *LNCS*, pages 182–194. Springer, 2010.

[12] C. Meadows. Analysis of the Internet Key Exchange protocol using the NRL Protocol Analyzer. In *IEEE Symposium on Security and Privacy*, pages 216–231, 1999.

[13] C. Meadows, P. F. Syverson, and I. Cervesato. Formal specification and analysis of the Group Domain Of Interpretation Protocol using NPATRL and the NRL Protocol Analyzer. *Journal of Computer Security*, 12(6):893–931, 2004.

[14] S. Meier, C. Cremers, and D. Basin. Efficient construction of machine-checked symbolic protocol security proofs. *Journal of Computer Security*, 21(1):41–87, 2013.

[15] S. Meier, B. Schmidt, C. Cremers, and D. Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In *Proc. CAV*, volume 8044 of *LNCS*, pages 696–701. Springer, 2013.

[16] B. Schmidt, S. Meier, C. Cremers, and D. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *Proceedings of the 25th IEEE Computer Security Foundations Symposium (CSF)*, pages 78–94, 2012.

[17] S. Xu and C.-T. Huang. Attacks on PKM Protocols of IEEE 802.16 and Its Later Versions. In *Wireless Communication Systems, 2006. ISWCS '06. 3rd International Symposium on*, pages 185 –189, sept. 2006.