Improving the Security of Cryptographic Protocol Standards

David Basin | ETH Zurich
Cas Cremers | University of Oxford
Kunihiko Miyazaki | Hitachi
Saša Radomirović | ETH Zurich
Dai Watanabe | Hitachi

Despite being carefully designed, cryptographic protocol standards often turn out to be flawed. Integrating unambiguous security properties, clear threat models, and formal methods into the standardization process can improve protocol security.

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 IPsec can be used to set up virtual private networks. These protocols are omnipresent and let us access and protect numerous applications, ranging from banking to social media. Many lesser-known protocols are also in use, such as WiMAX for secure communication in wireless networks, ISO/IEC 9798 for entity authentication, and the Extensible Authentication Protocol (EAP) for network access authentication.

A protocol such as TLS lets any client potentially 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 RFCs, which describe a protocol's operation in sufficient detail to guide the construction of interoperable implementations. All the protocols we have mentioned are described by standards or RFCs approved by standardization bodies, or are undergoing standardization.

A closer look at modern protocol standards indicates that although standardization bodies are doing excellent work, the resulting protocols' security varies considerably. Over the past decade, we have conducted numerous case studies with model-checking tools for security protocols, some of which we have developed ourselves.[1–4] Our analysis shows that many standards suffer from security weaknesses, including basic mistakes and well-known flaws. In some cases, these weaknesses have been quite serious. Even minor problems, however, are best avoided from the start, prior to standardization. Amending standards is time-consuming, and after amendment, companies with products implementing the standard must decide between costly upgrades or the risk of losing their reputation and undergoing litigation for distributing products with known defects.

Because experts carefully design standards, we might expect them to meet strong, well-understood, and well-specified security guarantees. Unfortunately, standards do not always meet this expectation. Although they often contain detailed functional descriptions, many do not include 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 past few decades, researchers have successfully used formal methods to analyze small, academic protocols with well-defined threat models (also

1540-7993/15/$31.00 © 2015 IEEE

The Ambiguity of Authentication

Authentication is a common security goal. However, the notion of authentication has numerous, substantially different interpretations, each with several variants. Table A presents three typical interpretations of "a client C authenticated by a server S," each with a weaker and a stronger variant.

Each of these interpretations has many more variants. The critical observation is that no one "right" definition of authentication exists: you cannot specify an appropriate authentication property without a fundamental understanding of the application scenario.

Table A. Typical interpretations of "a client C authenticated by a server S."

Variant	Entity authentication	Data agreement	Authenticated session key
Weaker	*Aliveness of C*: C has performed an action.	*Noninjective agreement on message m*: S has received the message *m* from C. C has sent *m* to S.	*Authenticated session key k*: session key *k* is a fresh session key, known only to C and S and possibly a trusted third party.
Stronger	*Recent aliveness of C*: C has performed an action (causally) after a specific action of S.	*Agreement on message m*: noninjective agreement on *m*, and S will not accept *m* if it is replayed by the adversary.	*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*.

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 as well as analyzing the standard with respect to properties not explicitly stated in the standard and therefore conjectured by the researchers.

Here, we illustrate the problems that arise when security properties and threat models are neglected in standards and present several case studies to demonstrate how formal methods can make a difference. We then examine how we might better integrate formal methods and associated tools into the standardization process given present obstacles and limitations. We base our case studies on three protocols: WiMAX, EAP, and ISO/IEC 9798.

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 (www.ieee802.org/16/published.html). 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, the privacy key management (PKM) protocols the standard describes.

IEEE originally proposed the WiMAX standard in 2001 and has updated it 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 as well as a security association identifier, which we denote by SAID in the following message exchanges for PKMv1-RSA:

SS ⟶ BS: SS_Certificate, SS_Algo_Suites, CID
BS ⟶ SS: $\text{Enc}_{PK(SS)}$(AK), SAID

After the standard's initial release, David Johnston and Jesse Walker identified several weaknesses in 2004.[5] In particular, they argued that PKMv1-RSA essentially provides no security guarantees because, in the context of wireless transmissions, we should assume that attackers can spoof arbitrary messages (that is, send messages impersonating another party). The subscriber station thus has no idea who encrypted or even generated the key it receives.

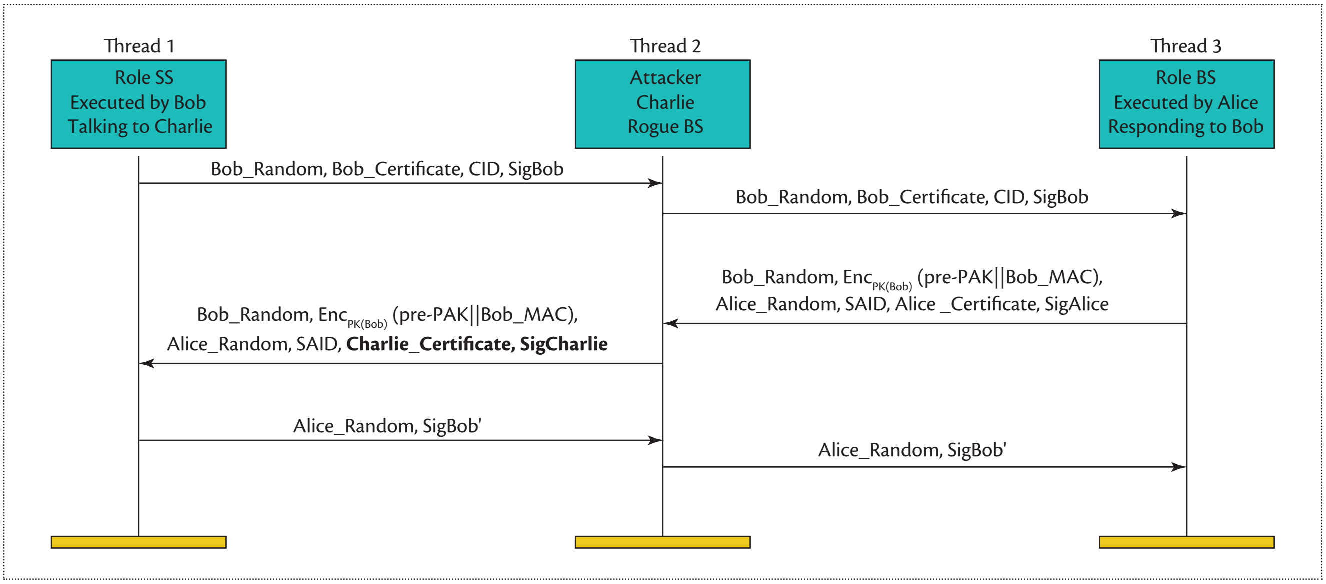
Thread 1
Role SS
Executed by Bob
Talking to Charlie
Thread 2
Attacker
Charlie
Rogue BS
Thread 3
Role BS
Executed by Alice
Responding to Bob
Bob_Random, Bob_Certificate, CID, SigBob
Bob_Random, Bob_Certificate, CID, SigBob
Bob_Random, Enc$_{PK(Bob)}$ (pre-PAK||Bob_MAC),
Alice_Random, SAID, Alice _Certificate, 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.

Johnston and Walker argued that 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, given that 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 PKMv1-RSA's weaknesses, "mutual authentication" is not a uniquely defined concept; authentication has many possible variations that differ in strength, as "The Ambiguity of Authentication" sidebar illustrates.

In 2005, IEEE released a new version of the standard that introduced the PKMv2-RSA protocol. This new version is a three-message protocol in which 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 this encrypted message back to the subscriber station 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′). We can see this in the following message exchanges for PKMv2-RSA:

SS → BS: SS_Random, SS_Certificate, CID, SigSS
BS → SS: SS_Random, Enc$_{PK(SS)}$ (pre-PAK||SS_MAC),
BS_Random, SAID, BS_Certificate, SigBS
SS → BS: BS_Random, SigSS′

It appears that this new protocol aimed to address the weaknesses in PKMv1-RSA. But again, the standard specified neither a threat model nor security properties. Consequently, even though the numbering might suggest that PKMv2-RSA provides properties in addition to those PKMv1-RSA provides, the standard offers no concrete statements to this effect.

Both academic and industrial experts were involved in a manual security review of drafts of the 2005 version of the standard.[6] 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" is possible on the PKMv2-RSA protocol.[7] This is a commonplace man-in-the-middle (MITM) attack in which the attacker forwards and selectively modifies messages between two parties. In 2008, Suzana Andova and her colleagues used the formal protocol analysis tool Scyther to analyze several subprotocols from the standard.[8] (The protocol models used in the analysis are available from https://github.com/cascremers/scyther/tree/master/gui/Protocols/IEEE-WIMAX.) We independently rediscovered the MITM attack, proposed a fix, and verified its correctness.

Figure 1 shows the attack, which 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 about who the base station is. Adding the base station's identity to the third message prevents the attack.[8]

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 the attack. So, is this "attack" really a security threat?

The surprisingly simple answer is that, because the standard specifies neither the intended security properties nor the threat model, we cannot know for sure. If we play it safe and assume that PKMv2-RSA does not provide strong security guarantees, then the cryptographic operations performed in it and the subsequent protocol are simply redundant overhead. In fact, we can discard PKMv2-RSA's third message without sacrificing the security properties that it achieves in composition with the subsequent protocol.[8] We could thus 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 MITM problem. However, this could lead to real problems if PKMv2-RSA is combined with a different subsequent protocol whose engineers rely on the statement in IEEE 802.15e that PKMv2-RSA achieves mutual authentication.

Unfortunately, this failure to specify the threat model and security properties is not an isolated case.

Extensible Authentication Protocol

Our second case study is EAP, developed by the Internet Engineering Task Force (IETF). Compared to other standardization 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 we refer to next, are freely available online (www.ietf.org). This lets us study the evolution of EAP, which is currently an IETF proposed standard (http://tools.ietf.org/html/rfc3748).

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 (Subscriber Identity Module), and EAP-AKA (Authentication and Key Agreement), used for authentication and session key distribution in Wi-Fi Protected Access (WPA/WPA2), the Global System for Mobile Communication (GSM), and the Universal Mobile Telecommunications System (UMTS) networks, respectively.

EAP began in 1995 as an Internet draft for the PPP Extensible Authentication Protocol. PPP is the Point-to-Point Protocol, first published as RFC 1134 in 1989. In April 2004, an Internet draft document was published reviewing 48 EAP authentication methods (http://tools.ietf.org/html/draft-bersani-eap-synthesis-sharedkeymethods-00). It concluded that some methods were no longer under active development, and many did not comply with the then-evolving EAP reference document, which became RFC 3748. Of the remaining methods, the Internet draft identified several interesting candidates but left their comparison for future work. A comparison at the time would have been difficult because an EAP threat model and specific security claims were only introduced in RFC 3748. In fact, even with RFC 3748's threat model and security claims, we still consider it a challenge to compare EAP authentication methods because the threat model is too vague.

This threat model is defined by the assumption that an attacker could compromise links over which EAP packets are transmitted, and by a list of 10 attacks. This is, of course, a source of ambiguity: any attack that is not explicitly mentioned could be considered out of the threat model's scope. Examining the 10 attacks more closely, we see that they mix generic attacker capabilities with specific 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 packet modification, but is restricted to EAP packets.

One way to obtain a more precise threat model is to focus on what we consider the essential attacker capabilities. From the first two items on the list, we infer that an attacker can eavesdrop on, spoof, and modify EAP packets. Several of the subsequent items consider specific attack scenarios that could result from these three capabilities. One concerns denial-of-service attacks by spoofing messages, and three others concern specific MITM attacks. The last item considers a particular scenario in which an attacker might 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 EAP's security properties. An EAP authentication method specification must state which security properties it claims to satisfy by referring to a nonexhaustive list given in section 7.2.1 of RFC 3748. RFC 3748 recommends that the claims be supported with evidence in the form of a proof or reference.

We examine a selection of properties relevant for making precise statements about a protocol's behavior. The property descriptions are lightly edited quotes from section 7.2.1:

- *Integrity protection* 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 protected fields.
- *Replay protection* refers to protection against the replay of an EAP method or its messages, including status messages.
- *Session independence* demonstrates that passive attacks (such as capturing the EAP conversation) or active attacks (including compromising the master session keys) do not enable the compromise of subsequent or prior keys.

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

Surprisingly, the confidentiality claim is based on a definition that unnecessarily complicates protocol analysis and comparison (see RFC 3748, section 7.3):

- *Confidentiality* refers to the encryption of EAP messages, including status indications and EAP requests and responses. A method making this claim must support identity protection.

There are two problems with this property. First, in an adversarially controlled network, encryption is necessary to ensure message confidentiality, but it is not sufficient in general. Danny Dolev and Andrew Yao constructed an artificial but striking example.[9] 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 provide not only message confidentiality but also "identity protection," a privacy feature that is an arguably unrelated property. The consequence of having these two distinct properties combined into one is that authentication methods that provide message confidentiality but not identity protection, such as EAP-PSK (Pre-Shared Key; RFC 4764), cannot be easily distinguished from authentication methods that provide neither of the two properties, such as EAP-MD5-Challenge (RFC 2284).

RFC 3748 has been updated with RFC 5247, in which the threat model is clearer, but the newer version does not update the security claims. Still, there is clear movement toward a more precise security model. Moreover, an IETF best current practices document published in 2007 (RFC 4962) advocates using 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.

ISO/IEC 9798

The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) jointly develop IT standards. In 1991, they published the first part of our final case study, ISO/IEC 9798, 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, ISO/IEC has revised parts of the standard several times to address weaknesses and ambiguities. We might thus expect that such a mature and pervasive standard is "bulletproof" and that the protocols satisfy strong, practically relevant authentication properties.

However, it is not entirely clear which security properties the standard's protocols provide. The standard claims to provide "entity authentication," alternatively phrased as "authentication of the claimant identity." As the sidebar explains, we can interpret the notion of authentication in different ways, making it extremely difficult for users to judge if a particular protocol provides a sufficient form of authentication. Similarly, as is

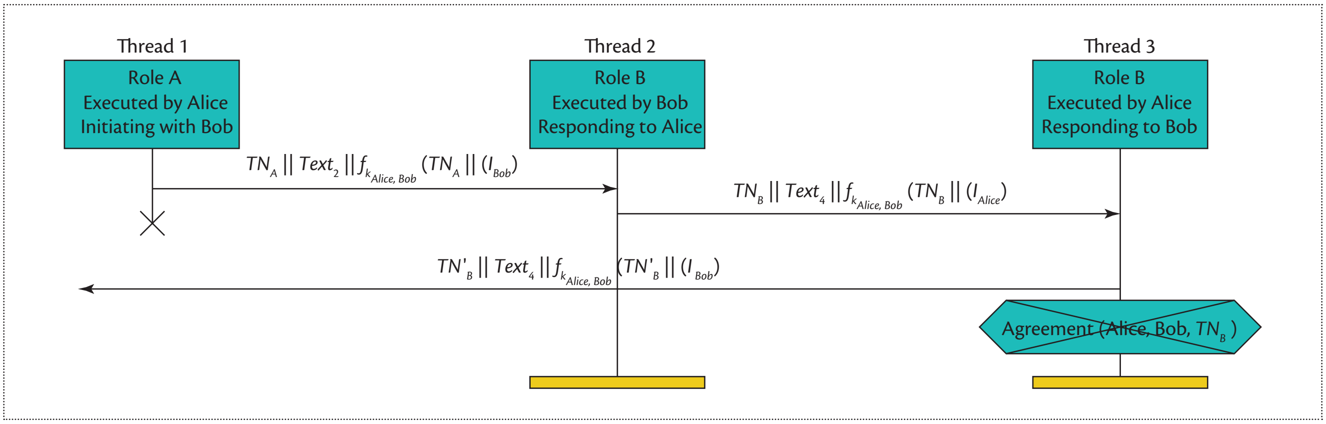
Thread 1
Role A
Executed by Alice
Initiating with Bob
Thread 2
Role B
Executed by Bob
Responding to Alice
Thread 3
Role B
Executed by Alice
Responding to Bob
$TN_A \,||\, Text_2 \,||\, f_{k_{Alice,\,Bob}}(TN_A \,||\, (I_{Bob})$
$TN_B \,||\, Text_4 \,||\, f_{k_{Alice,\,Bob}}(TN_B \,||\, (I_{Alice})$
$TN'_B \,||\, Text_4 \,||\, f_{k_{Alice,\,Bob}}(TN'_B \,||\, (I_{Bob})$
~~Agreement (Alice, Bob, TN_B)~~

Figure 2. Role mix-up attack. This attack occurs 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.

common in many standards, the threat model is defined only in terms of specific (informal) attack types, such as "replay attack."

We became involved in evaluating ISO/IEC 9798 in 2010 with the Cryptography Research and Evaluation Committee set up by the Japanese government. We formally analyzed the 2010 versions of the protocols specified, parts 1–4 of ISO/IEC 9798, using the Scyther tool.[2] 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 previously defined authentication properties.[10]

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

Figure 2 shows an example of a role-mixup attack on the following, 2009 version of the two-pass mutual authentication protocol using a cryptographic check function:

$$A \rightarrow B: TN_A||Text_2||f_{K_{AB}}(TN_A||I_B||Text_1)$$
$$B \rightarrow A: TN_B||Text_4||f_{K_{AB}}(TN_B||I_A||Text_3)$$

Agents perform actions such as sending and receiving messages, resulting in message transmissions (horizontal arrows). Actions are executed in threads (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.

In addition, when a protocol implementation uses the optional text fields *Text1* and *Text3*, 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 *Text1*, whereas Bob actually sent this data in the *Text3* field. Depending on how these fields are used, this could 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 *Text1* represents the amount of money to be transferred. Assume the second message is used for confirmation, where *Text3* corresponds to the transaction number. In this case, the adversary can reuse 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, as well as agents' willingness to engage in the protocol.

We analyzed the shortcomings in the protocols' design and proposed and formally verified repairs.[11] 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 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 (finding errors in the early phases) and verification (providing objective and verifiable security guarantees during end phases).

Discussion and Recommendations

Our three case studies suggest a trend toward 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 in which 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 decide whether a protocol is suitable for a given purpose. The ISO/IEC 9798 case study demonstrates that a standard can provide systematic threat models and precise security properties and that we can perform formal verification. It also shows that formal methods are slowly starting to affect standardization bodies.[11–13] We expect this trend to continue as governments and other organizations increasingly push for the use of formal methods in developing and evaluating critical standards.

For example, in 2007, ISO/IEC JTC 1/SC 27 (IT Security Techniques) started the "verification of cryptographic protocols" project, which involves developing a standard (ISO/IEC 29128) for certifying cryptographic protocol designs in which the highest evaluation levels require using formal, machine-checked correctness proofs.[14] The four cornerstones of the ISO/IEC 29128 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.[15] The specifics for these requirements depend on what protocol assurance level (PAL) is sought. At the lowest assurance level, PAL1, informal descriptions might be given for the protocol specification, adversary model, and security properties. The self-assessment can 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 satisfy these requirements 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. We must bridge several gaps 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. Others are in our own backyard and concern the automated tools employed in the verification process. For widespread industrial use, these tools must be robust, well-documented, and go beyond current research prototypes. Moreover, the tools themselves must ultimately be certified to be correct.

Much work remains 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 different perspectives. Addressing this requires research leading to a common framework. Ideally, we need a standardized set of unambiguous security properties and threat models that other standards can refer to. Once standards add these to their functional specifications, we will have the foundation for evaluating standards' security merits, and subsequently for comparing different proposals, possibly using tool support.[2,16,17] ■

References

1. D. Basin, S. Mödersheim, and L. Viganò, "OFMC: A Symbolic Model Checker for Security Protocols," *Int'l J. Information Security*, vol. 4, no. 3, 2005, pp. 181–208.
2. C. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols," *Proc. 20th Int'l Conf. Computer Aided Verification*, LNCS 5123, 2008, pp. 414–418; www.cs.ox.ac.uk/people/cas.cremers/scyther.
3. S. Meier, C. Cremers, and D. Basin, "Efficient Construction of Machine-Checked Symbolic Protocol Security Proofs," *J. Computer Security*, vol. 21, no. 1, 2013, pp. 41–87.
4. B. Schmidt et al., "Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties," *Proc. 25th IEEE Computer Security Foundations Symp.* (CSF), 2012, pp. 78–94.
5. D. Johnston and J. Walker, "Overview of IEEE 802.16 Security," *IEEE Security & Privacy*, vol. 2, no. 3, 2004, pp. 40–48.

6. B. Aboba, "Summary of the IEEE 802.16e D8 Security Review," Sept. 2005; www.ieee802.org/16/tge/contrib/C80216e-05_373.pdf.
7. S. Xu and C.-T. Huang, "Attacks on PKM Protocols of IEEE 802.16 and Its Later Versions," *Proc. 3rd Int'l Symp. Wireless Communication Systems*, 2006, pp. 185–189.
8. S. Andova et al., "A Framework for Compositional Verification of Security Protocols," *Information and Computation*, Feb. 2008, pp. 425–459.
9. D. Dolev and A. Yao, "On the Security of Public Key Protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, 1983, pp. 198–208.
10. G. Lowe, "A Hierarchy of Authentication Specifications," *Proc. 10th IEEE Computer Security Foundations Workshop* (CSFW), 1997, pp. 31–44.
11. D. Basin, C. Cremers, and S. Meier, "Provably Repairing the ISO/IEC 9798 Standard for Entity Authentication," Proc. *1st Int'l Conf. Principles of Security and Trust* (POST), LNCS 7215, P. Degano and J.D. Guttman, eds., 2012, pp. 129–148.
12. C. Meadows, "Analysis of the Internet Key Exchange Protocol Using the NRL Protocol Analyzer," *Proc. IEEE Symp. Security and Privacy*, 1999, pp. 216–231.
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," *J. Computer Security*, vol. 12, no. 6, 2004, pp. 893–931.
14. S. Matsuo et al., "How to Evaluate the Security of Real-Life Cryptographic Protocols? The Cases of ISO/IEC 29128 and CRYPTREC," *Proc. 14th Int'l Conf. Financial Cryptography and Data Security*, LNCS 6054, 2010, pp. 182–194.
15. ISO/IEC 29128: Information Technology—Security Techniques—Verification of Cryptographic Protocols, Int'l Organization for Standardization, 2011.
16. B. Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules," *Proc. 14th IEEE Computer Security Foundations Workshop* (CSFW), 2001, pp. 82–96.
17. S. Meier et al., "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," *Proc. 25th Int'l Conf. Computer Aided Verification* (CAV), LNCS 8044, 2013, pp. 696–701.

David Basin is a full professor and the Information Security Chair in the Department of Computer Science at ETH Zurich. His research focuses on information security, in particular, methods and tools for modeling, building, and validating secure and reliable systems. Basin received a Habilitation in computer science from the University of Saarbrücken. Contact him at basin@inf.ethz.ch.

Cas Cremers is an associate professor in the Department of Computer Science at the University of Oxford. His research focuses on information security and applied cryptography, including the development of automated analysis tools. Cremers received a PhD in computer science from Eindhoven University of Technology. Contact him at cas.cremers@cs.ox.ac.uk.

Kunihiko Miyazaki is a senior researcher in the Research and Development Group at Hitachi. His research interests include information security, cryptography, and formal methods. Miyazaki received a PhD in information and communication engineering from the University of Tokyo. He's a member of the Information Processing Society of Japan and the Institute of Electronics, Information, and Communication Engineers. Contact him at kunihiko.miyazaki.zt@hitachi.com.

Saša Radomirović is a senior scientist in the Department of Computer Science at ETH Zurich. His research focuses on information security, in particular, modeling, analysis, and verification of security and privacy properties with algebraic and combinatorial methods. Radomirović received a PhD in mathematics from Rutgers University. Contact him at sasa.radomirovic@inf.ethz.ch.

Dai Watanabe is a senior researcher in the Research and Development Group at Hitachi. His interests include information security, cryptography, and cryptographic protocols. Watanabe received a doctorate in engineering from the Tokyo University of Science. He's a member of the Information Processing Society of Japan and the Institute of Electronics, Information, and Communication Engineers. Contact him at dai.watanabe.td@hitachi.com.