Provable Security in the Real World

Provable security plays an important role in the design and analysis of systems using cryptography. However, protocols can be vulnerable to attacks outside the scope of the existing formal analyses.

f we define science as an objective approach to analysis and pursuit of knowledge on the basis of rigorous logical arguments, then cryptography undoubtedly has become a science. The area of research responsible for bringing about this transition is commonly known as provable security. Provable security introduces formal definitions of security and adopts techniques from probability theory and computational complexity theory to analyze the security of cryptographic constructs. Recently, the merits of this approach have been subject to debate, spurred in part by a series of articles by Neal Koblitz and Alfred Menezes, beginning with "Another Look at 'Provable Security."¹ One of the main issues at stake here is the degree of assurance that provable security provides. Researchers have discovered many attacks on cryptographic schemes that were previously proven to be secure. However, we must consider that no science provides an absolute guarantee of the truth of its results, and that sciences evolve over time. Indeed, compared to more mature sciences, the field of cryptography, while developing rapidly, is still new. Moreover, the application of any scientific discipline always lags behind theoretical developments. As Einstein is reputed to have said, "In theory, theory and practice are the same. In practice they are not." In this article, we illustrate some of the gaps that exist between cryptography's theory and practice (see the "Cultures in Cryptography" sidebar).

Although researchers have applied provable security to almost all aspects of cryptography, we focus only on its application to analyzing the security of symmetric encryption schemes used in Internet Protocol security (IPsec), Secure Sockets Layer (SSL)/ Transport Layer Security (TLS),

and Secure Shell (SSH) protocols. However, much of what we say here applies more widely to other aspects of cryptography and analysis techniques, such as formal methods and universal composability.

Two Milestones in Provable Security

Provable security dates back to 1949 when Claude Shannon used information-theoretic concepts to prove the perfect secrecy of the one-time pad in a paper titled "A Mathematical Theory of Cryptography."² Perfect secrecy is an information theoretic notion of security that, informally, defines an encryption scheme to be perfectly secure if the ciphertext doesn't leak information about the plaintext. Note that perfect secrecy makes no assumptions about the adversary's computational capabilities, meaning that an exhaustive search on the encryption key would convey no additional information. The drawback of a perfectly secret encryption scheme is that its key space needs to be at least as large as its plaintext space. This creates significant issues for the practical deployment of such a scheme. Essentially, the problem of secure communication is replaced by the problem of generating, distributing, and securely destroying the large amounts of keying material necessary for the scheme. Indeed, such schemes haven't been widely deployed despite their proven security properties.

In 1984, Shafi Goldwasser and Silvio Micali made



JEAN PAUL DEGABRIELE AND KENNETH G. PATERSON *Royal* Holloway, University of London

GAVEN J. WATSON University of Calgary

Cultures in Cryptography

t might seem strange that there can be such an obvious discrepancy between the theory and practice of cryptography. This gap might be in part due to the fact that different actors approach the field from completely different perspectives.

Theoreticians

First, consider a typical theoretical cryptographer's viewpoint. Judging by cryptographic research literature, one might surmise that many in the community view cryptography as a branch of theoretical computer science. With notable exceptions, the literature addresses theoretical questions as opposed to the real-world problems currently affecting widely used protocols. Although this focused approach has driven the field forward as a theoretical subject, it has done less to improve the security of currently used systems and makes the field seem less accessible to those in the applied community.

Practitioners

Provable security has become an important research area in modern cryptography but is still met with skepticism by many in the practical community. This is understandable considering the types of attack that we outline here. Practitioners might think provable security results provide an absolute statement of

> a breakthrough in provable security, introducing semantic security for public-key encryption.³ Semantic security adapts Shannon's perfect secrecy to the computational setting, considering only adversaries having bounded computational resources. Informally, semantic security states that no computationally bounded adversary given the ciphertext can predict anything about the plaintext any better than it can when it's not given the ciphertext. Goldwasser and Micali's original scheme was very inefficient in terms of computation and bandwidth, involving bit-by-bit encryption of plaintexts. However, we now have schemes that achieve semantic security efficiently, with security based on reasonable number-theoretic hardness assumptions. Thus, although weaker than perfect secrecy, semantic security doesn't pose the same practical limitations as perfect secrecy. Goldwasser and Micali's introduction of semantic security is widely recognized as the start of provable security as we know it today.

Understanding Security Proofs

Provable security, as its name implies, is about proving cryptographic schemes secure according to some notions of security. Such proofs commonly take the form of *reductions*, a concept borrowed from complexity theory. Reducing problem *A* to problem *B* means

security, especially if they're presented in such a manner. When they later discover that a scheme is insecure because of an attack outside the security model, this might damage their confidence in the whole enterprise of provable security.

In the practical community, there are at least two viewpoints to consider: the perspective of specification document writers and that of the implementers using these documents as a guide for writing code. Usually, writers must include some flexibility in a specification to allow for differences in implementations, allow interoperability, and account for the competing interests of the different parties contributing to the development process. Also, there's often a requirement to support certain options for backward compatibility. Unfortunately, this flexibility only increases the specifications' complexity and the consequent risk that an unfortunate feature interaction will lead to an attack.

Next, consider the practitioner implementing a system using cryptography. Because implementers base their implementations on the relevant specification documents and RFCs, they might not read theoretical research papers. With many options open to implementers in these documents, they might introduce a vulnerability. Without knowledge of the theory behind the choice being presented, implementers will have little idea of the effect a particular choice might have on security.

> that given access to an algorithm M_B that solves problem B, we can construct an algorithm M_A that employs M_B as a subroutine to efficiently solve problem A. Now suppose we can reduce solving problem Pto breaking an encryption scheme E. Then, by contraposition, breaking E is at least as hard as solving P. Such a reduction would then represent a *conditional* proof of security for encryption scheme E on the basis that problem P is hard. However, we still need to clarify *break* and *hard*. We illustrate these concepts in the context of encryption schemes.

> A break is normally defined through a game, in which one player (the challenger) gives well-defined capabilities to the other player (the adversary), and the adversary winning the game directly translates to a breach of a well-defined and well-motivated security notion. Semantic security for public-key encryption is an instance of such a notion but is cumbersome to use in proofs. Therefore, proofs often use the notion of *indistinguishability* (IND). Goldwasser and Micali showed that indistinguishability implies semantic security, and in "The Notion of Security for Probabilistic Cryptosystems," Micali and colleagues later proved the converse, showing that the two notions are equivalent (at least in the framework of polynomial-time reductions).⁴

The IND game for public-key encryption has two

phases. In the first phase, the challenger generates a public/private key pair for the scheme and gives the adversary the public key. The adversary performs its computation, selects two equal-length plaintexts m_0 , m_1 , and submits these to the challenger. The challenger randomly picks one of the two plaintexts, encrypts it, and gives the resulting challenge ciphertext to the adversary. In the second phase, the adversary performs further computation and then guesses which of its two chosen plaintexts was encrypted in the challenge ciphertext. The adversary wins the IND game if it guesses correctly. Because an adversary can always win this game with probability 1/2 by guessing randomly, we consider the adversary successful (and the scheme broken) only if it wins the IND game with probability significantly greater than 1/2. Notice that because the adversary has access to the public key, it could simply reencrypt the two plaintexts m_0 , m_1 in the second phase. If the encryption scheme was deterministic, then one of the two results would match the challenge ciphertext, and the adversary would win the IND game with probability 1. Thus, a semantically secure encryption scheme can't be deterministic.

Having described the adversary's goal in the IND game, we next classify its capabilities more carefully. Throughout our game, the adversary has the public key and can perform arbitrary encryptions. In the symmetric-key setting, we can't simply give the adversary the key, so we give the adversary access to an encryption oracle. The adversary submits arbitrary plaintexts to this oracle and in return receives ciphertexts encrypting those plaintexts. In public-key and symmetric-key settings, we then talk of a chosen plaintext attack (CPA). To capture a more powerful adversary, we also consider the chosen ciphertext attack (CCA) setting. Here, we give the adversary access to both the public key (or an encryption oracle for the symmetric-key setting) and a decryption oracle. The adversary sends ciphertext to the decryption oracle and in return obtains that ciphertext's decryption, or an indication that the decryption process failed. (However, we can't let the adversary query the challenge ciphertext to the decryption oracle, or winning the IND game would be trivial.) This oracle models the fact that, in practice, an adversary might have access to such a decryption capability, for example, through interaction with the scheme's users. It also reflects a general conservatism in provable security: to capture the widest set of possible attacks in our model, we try to make attackers as powerful as possible (without making it trivial for them to win). In addition to being given access to the oracles, the adversary can perform arbitrary computations. In this way, a security proof applies to all adversaries, not simply those

who behave in particular ways that might be limited by designers' imaginations.

However, we must limit the amount of resources consumed by the adversary. Otherwise, for example, the adversary could simply perform an exhaustive search for the private key, rendering our definitions vacuous. We use complexity theory and the framework of polynomial-time algorithms to assist us here. We introduce a security parameter k that can be thought of as defining the key size in the encryption scheme. Then an encryption scheme is said to be IND-CPA or IND-CCA secure if we can prove that no polynomial-time-bounded adversary can win the respective IND game with probability that is nonnegligibly greater than $\frac{1}{2}$ in the parameter k. Here, a function f(k) is said to be negligible if for every polynomial p(k) there exists an N such that for all integers k > N, f(k) < 1/p(k). Thus, a non-negligible function is one that grows faster than the inverse of some polynomial in k, and the scheme is secure if no polynomial-time adversary has an advantage that grows in this way.

The proof of such a statement takes the form of a reduction from a problem P to the problem of winning the particular scheme's IND-CPA or IND-CCA game with non-negligible probability. However, barring a major breakthrough in complexity theory, the existence of cryptographically useful problems P not having polynomial-time solutions remains in doubt. We can only prove our cryptographic scheme's security on the basis of the assumption that P really is sufficiently hard. Hence, the security proof is only conditional. For this reason, some consider the term provable security misleading or inappropriate and prefer the term reductionist security. Nonetheless, this approach is beneficial for several reasons. First, for many years, researchers have analyzed certain computational problems, such as integer factorization, the discrete logarithm problem over a finite field, and the problem of finding the shortest vector in a lattice, and these are now generally considered to be hard. More specifically, for certain choices of parameters, we can estimate the success probabilities and running times of algorithms for solving such problems. When a new cryptographic scheme is introduced, reducing a well-known and hard problem to its security transfers our confidence about the problem's hardness to the cryptographic scheme's security. Furthermore, because we can compare and contrast the hardness of problems through their complexity classes, we can use this as a criterion for comparing the security of cryptographic schemes.

Concrete security is a natural development of the above approach. Instead of working in the fairly abstract framework of polynomial-time algorithms and negligible functions, we try to directly relate any ad-

versary's success probability and running time to that of the algorithms to solve some assumed-to-be-hard problem P via the security reduction obtained in the proof. For example, suppose we can prove that for any adversary running in time t and having success probability $1/2 + \varepsilon$ against the scheme, there's an algorithm to solve problem P that runs in time $f(t, \varepsilon)$ and has success probability $g(t, \varepsilon)$, for functions f and g. Then, by inverting this relationship, we can in principle use our knowledge about the current and projected state of the art in solving problem P along with our desired security level to select a concrete security parameter for the scheme. This approach also provides a means by which to judge the quality of proofs: a better proof is one that closely relates the resources an adversary consumed when breaking the scheme to those of an algorithm to solve the underlying problem P.

Another benefit of the provable security approach is that it enables the analysis of complex cryptographic schemes and protocols in terms of the security of simpler cryptographic primitives from which they're constructed. Such constructions might achieve more useful goals or meet stronger notions of security than their constituent components. By reducing a constructed scheme's security to that of its component primitives, we can achieve a higher security goal without introducing any further assumptions. Mihir Bellare and Chanathip Namprempre offered one such construction.⁵ They considered the sequential composition of a symmetric encryption scheme that is IND-CPA secure and a MAC (message authentication code) that is SUF-CMA (strongly unforgeable under chosen message attack) secure. Here, SUF-CMA is a standard security notion for a MAC. Bellare and Namprempre showed that such an encrypt-then-MAC construction yields an IND-CCA-secure encryption scheme. Hugo Krawczyk showed similar results for the analogous MAC-then-encrypt construction, for certain classes of encryption scheme.⁶

Some Limitations of Security Proofs

Provable security has obvious applications to cryptographic practice, such as protocol design and scheme selection. The existence of a correct security proof should be an important factor in selecting cryptographic schemes and adopting constructions for use in real-life protocols. However, we should take care when interpreting provable security results for practice. To reiterate, a security proof isn't an absolute guarantee of security. Rather, proofs are conditional, and security is guaranteed only as long as the underlying assumptions hold. But the reasons why caution is needed extend beyond this point.

In particular, security claims might not completely capture what should be considered a break in real life. For instance, indistinguishability implies that the adversary can't distinguish between the encryptions of plaintexts of its choice, given that they're of equal length. However, in real life, an attacker might be able to predict something about the plaintext through the ciphertext's length unless we take extra precautions to prevent such traffic analysis. In some applications, this might constitute a confidentiality breach (see "Attack-ing and Repairing the WinZip Encryption Scheme" for a concrete example in the context of data compression⁷). In addition, the basic indistinguishability definition says nothing about denial of service or attacks based on message replays. Thus, we must be aware of what security is guaranteed by the proof as opposed to what our security requirements are.

Furthermore, the capabilities we gave the adversary in the security game might not accurately reflect those a real-world adversary possesses. We offer several examples of practical attacks in which adversaries can glean small amounts of information about the plaintext format from the manner in which the decryption process proceeds (or fails to proceed). In some cases, this information can be leveraged to enable real-world adversaries to extract plaintext from a challenge ciphertext. This information is rarely included in formal security models, but our examples show the importance of considering this kind of leakage. More generally, side-channel attacks use extra information produced by an implementation to mount an attack. These types of attacks might not be captured by traditional formal security analyses because they lie outside the scope of the security model. To obtain meaningful real-world security guarantees from a security proof, it's essential that the security model used accurately captures all the powers available to real-world adversaries.

Furthermore, designing and implementing secure cryptosystems—and combining various cryptographic primitives—are difficult tasks with many different aspects to consider. Vulnerabilities might be introduced inadvertently because of a fundamental system design flaw or by a programmer during implementation. Specification documents, if available, typically give implementers some flexibility, but without proper and detailed guidance, their choices can affect a system's security dramatically. Even a slight change to a provably secure scheme can render the system insecure. So how do we know if an implementation is an accurate translation of what has been modeled and proven secure?

Applying Provable Security to Secure Communications Protocols

Here, we highlight problems that arise when applying provable security results. Our examples focus on symmetric cryptography as used in SSL/TLS, IPsec, and SSH. These examples make good case studies because the protocols are widely deployed and appear to be simple enough to be amenable to provable security analysis.

SSL/TLS

SSL is perhaps the most popular secure network protocol. One of its main uses is to secure credit-card data for Internet purchases. SSL was originally developed in the mid 1990s by Netscape and has since been adopted by the Internet Engineering Task Force (IETF) and renamed TLS. The current version of the protocol, TLS 1.2, is defined in RFC 5246. SSL/TLS consists of a number of subprotocols; here, we focus on the SSL/TLS Record Protocol, which provides confidentiality and integrity services for all SSL/TLS messages. At a high level, the Record Protocol uses a MAC-then-encrypt construction. This means that the message to be sent first has a MAC calculated on it and then encryption is performed over the concatenation of the message and the MAC tag. In addition, if a block cipher in CBC (cipher block chaining) mode is used, then the RFC specifies that we must add padding after the MAC is added and before the encryption. The padding must follow a particular format, and the RFC requires that this format be checked again after decryption.

Now consider what's known about encryption schemes following a MAC-then-encrypt structure from a provable security perspective. Bellare and Namprempre performed a formal analysis of generic compositions of a MAC and an IND-CPA-secure symmetric encryption scheme.⁵ They studied three constructions in detail: encrypt-and-MAC, encryptthen-MAC, and MAC-then-encrypt. Of these, they showed that encrypt-then-MAC is the only one that can achieve IND-CCA security (starting with appropriate notions of security for the encryption and MAC components). Indeed, they showed that the MACthen-encrypt construction isn't IND-CCA secure by exhibiting an SUF-CMA-secure MAC scheme and an IND-CPA secure encryption scheme whose MAC-then-encrypt composition can be broken in the IND-CCA attack model. Their analysis tells us nothing positive about the specific MAC-thenencrypt case used in SSL/TLS. Krawczyk further analyzed the MAC-then-encrypt composition, showing that if we use CBC mode of a good block cipher as the encryption component, then a MAC-thenencrypt construction will provide a secure channel.⁶ A secure channel is weaker than IND-CCA security but provides a positive result about the construction's security. In fact, Krawczyk's results can be improved to show that MAC-then-encrypt achieves IND-CCA

security when the encryption component is instantiated using CBC mode of a good block cipher.

So we now have a security proof for the MACthen-encrypt construction using CBC mode. Is this enough to claim security of the SSL/TLS Record Protocol in practice? Even setting aside implementation questions, the answer, unfortunately, is no. Brice Canvel and colleagues' attack against the MAC-thenencrypt construction used in SSL/TLS worked in practice against OpenSSL, one of the most widely used implementations.⁸ Their attack exploits the way SSL/TLS handles padding during encryption. Recall that when SSL/TLS uses CBC mode, some padding must be added to the message before encryption. This padding's format should also be checked during decryption, with an error message returned over the Record Protocol if the format is incorrect. Similarly, an error message is returned if the MAC fails to verify. These errors are "fatal"-the SSL/TLS channel is destroyed if either of the errors arises. Because of the construction, it's natural to check the padding before the MAC when decrypting. Canvel and colleagues' attack exploits the small timing difference that arises between the appearance of a padding error and MAC error to mount a special type of side-channel attack called a *padding oracle attack*. This type of attack can recover plaintext from arbitrary ciphertext blocks. Moreover, if a fixed plaintext is repeated across many sessions (for example, a password), then the attack can be iterated to boost its success probability even though the channel is destroyed at each attempt.

Canvel and colleagues' attack seems to contradict Krawczyk's security proof. How does the contradiction arise? Krawczyk's proof is mathematically correct, but his model doesn't accurately capture the way SSL/TLS works in practice. Indeed, the particular construction that Krawczyk studied doesn't include padding: it simply assumes that the plaintext lengths are already suitable for applying a block cipher in CBC mode. Also, his analysis doesn't consider the possibility of distinguishable error outputs. This means that although Krawczyk's analysis is correct and rules out many possible attacks against SSL/TLS, it can't be applied directly to the protocol as specified. Instead, we need a more accurate model that reflects the fact that SSL/TLS really uses a MAC-then-pad-then-encrypt construction. Kenneth Paterson and Gaven Watson have taken initial steps in this direction in "Immunising CBC Mode against Padding Oracle Attacks: A Formal Security Treatment," in which they study the CBC mode's IND-CPA security in a situation in which the attacker has access to padding error information.⁹ However, extending this type of analysis to the IND-CCA case for the specific construction used in SSL/TLS remains an open problem. We still don't have strong, formally proven guarantees for the SSL/ TLS Record Protocol.

Note that the IETF community's reaction to Canvel and colleagues' SSL/TLS attacks wasn't to change the protocol to use a construction known to be IND-CCA secure in the face of padding oracle attacks (for example, by using a different padding scheme or by using a different construction entirely). Instead, RFC 5246 recommends that implementers ensure the MAC and padding errors aren't distinguishable by content or timing. This prevents the attack and requires minimal code changes. Thus, a pragmatic but unproven fix has been the solution of choice, reflecting the constraints and compatibility issues the IETF community faces when changing such a significant protocol.

IPsec

IPsec is a protocol suite offering security services at the TCP/IP stack's IP layer. IPsec is most widely deployed to build virtual private networks (VPNs) and secure remote access solutions. Its major components are the AH (authentication header), ESP (encapsulating security payload), and IKE (Internet key exchange) protocols specified in RFCs 4302, 4303, and 4306, respectively. AH provides data-origin authentication and replay protection services using a MAC in combination with sequence numbers and a replay window. ESP originally provided a data confidentiality service but now provides data-origin authentication as well. IKE provides an automated security capability negotiation and key-management service. Because of IPsec's flexibility, users have various options for how to combine these protocols to build a secure channel that suits their needs.

Jean Paul Degabriele and Kenneth Paterson recently demonstrated a series of plaintext-recovery attacks against various IPsec configurations in which AH is first applied to an IP datagram and the result is then protected by ESP in encryption-only mode.¹⁰ This is yet another instantiation of the MAC-then-encrypt construction, analyzed by Bellare and Namprempre⁵ and Krawczyk.⁶ These attacks work regardless of the block cipher or MAC algorithm, as long as we use the block cipher in CBC mode. As with Canvel and colleagues' SSL/TLS attack, Degabriele and Paterson's attacks exploit several features from the IPsec realization of the basic MAC-then-encrypt composition, which aren't captured in Krawczyk's security model. These include fields in the AH-protected packet that aren't covered by the MAC algorithm, extra unauthenticated padding fields in the plaintext before encryption (namely, traffic flow confidentiality [TFC] padding bytes, encryption padding bytes, and the next-header byte), and the fact that IPsec should check certain padding fields for correctness after decryption. Here, incorrect formatting leads to a packet drop, whereas correct formatting leads to further processing, which can be arranged to result in a response message being sent on the secure channel. The presence (or absence) of such a message can reveal a small amount of information about the plaintext format, which can in turn be leveraged to recover arbitrary amounts of plaintext in a reliable manner. One of Degabriele and Paterson's attacks also exploits the manner in which IPsec's processing interacts with IP fragment processing to generate error messages that the attacker can detect.

These attacks on IPsec—as with those on SSL/ TLS—demonstrate that an attacker can exploit error information not normally considered in formal analyses to mount attacks against provably secure cryptosystems. Formal analyses that don't consider such error information will typically fail to capture all the subtleties in the implementation of a secure channel protocol, and this can lead to attacks.

Another important distinction between Krawczyk's theoretical model and the way in which cryptography operates in IPsec relates to atomicity. Krawczyk's model treats the MAC-then-encrypt construction as an atomic operation (although a counterexample given in his article highlights that MAC-then-encrypt might not be secure when an intermediate encoding step is introduced). However, in IPsec, the MAC-then-encrypt configurations are realized by combining two separate protocols, AH and ESP, with each protocol performing its own processing, and with the possibility of additional processing between these steps.

SSH

Finnish researcher Tatu Ylönen originally designed SSH in the mid 1990s to replace insecure remote logins after his university's network was targeted by some password-sniffing attacks. In 2006, a collection of RFCs specified SSH version 2. We're interested in RFC 4253, which specifies the SSH Binary Packet Protocol (BPP)-the part of SSH responsible for ensuring messages' confidentiality and integrity. The BPP follows an encode-then-encrypt-and-MAC construction. The payload message is first encoded and then encrypted; the MAC is calculated on the encoded message together with a sequence number. A ciphertext packet consists of the concatenation of the encrypted message and the MAC tag. The encoding scheme adds three fields to the payload message: a packet-length field, a padding-length field, and some padding bytes. The packet-length field specifies the total length of what follows, that is, the combined length of the padding-length field, the payload message, and the padding field. It's ostensibly encrypted

to protect against traffic analysis. This has a significant effect on the protocol's security.

Mihir Bellare and colleagues performed a formal analysis of the SSH BPP.¹¹ Because of a distinguishing attack by Wei Dai that exploits the use of initial packet chaining when using CBC mode encryption,¹² Bellare and colleagues weren't able to directly prove security for the SSH BPP as defined in RFC 4253. Instead, they proposed several minor SSH BPP variants and proved them secure in an extended version of the IND-CCA model, with the new model taking into account the stateful nature of SSH decryption.

As with our previous examples, the SSH BPP turned out to be vulnerable to attack, despite having a mathematically correct security proof. Martin Albrecht and colleagues discovered plaintext-recovery attacks that exploit the SSH BPP's use of an encrypted packet-length field, its reliance on CBC mode, and the attacker's ability to send ciphertext data in small chunks and observe how the recipient reacts.¹³ The attack principles are simple. An attacker observes a ciphertext and chooses one block to attack. The attacker sends this target block to the recipient in such a way that the recipient interprets it to be the start of a new packet. The recipient must immediately decrypt this block to retrieve the packet-length field, to know how much data it must wait for before it receives and verifies the MAC. The attacker then proceeds by sending random blocks one at a time until the recipient outputs a MAC error. By counting how many random blocks have been sent, the attacker can deduce the new packet's packet-length field and, by the properties of CBC mode, deduce the corresponding bits in the target plaintext block. In practice, this attack is complicated by checks performed on the packet-length field once the recipient covers it. Albrecht and colleagues implemented variants of this attack against OpenSSH, one of which recovers 32 bits of plaintext with probability 2^{-18} .

This attack can be applied to one of the provably secure variants of the SSH BPP proposed by Bellare and colleagues. So what went wrong? First, their analysis assumes that ciphertexts are self-describing in terms of their lengths. In reality, we see that recipients must decrypt the first block of a packet as soon as they receive it to obtain the packet length. RFC 4253 actually states that

[i]mplementations SHOULD decrypt the length after receiving the first 8 (or cipher block size, whichever is larger) bytes of a packet.

In addition, Bellare and colleagues' model doesn't allow for the possibility that the amount of data needed to complete the decryption process is governed by data produced *during* the decryption process. Second, we note that in their analysis, ciphertexts and plaintexts are handled as atomic strings. In contrast, Albrecht and colleagues' attacks exploit the fact that an attacker can send data in small chunks to the recipient. Many implementations use a buffer to store data until it's needed, but Bellare and colleagues' analysis doesn't model this.

Kenneth Paterson and Gaven Watson recently provided a new formal analysis of the SSH BPP using counter-mode encryption, with the explicit intention of addressing the shortcomings of the previous analysis.¹⁴ Paterson and Watson begin by defining a new version of SSH-CTR that accurately captures how the SSH BPP with counter-mode encryption is defined in the RFCs and coded in practice in OpenSSH and other implementations. They also extended the previous security model to account for the manner in which the SSH BPP buffers as-yet-unprocessed ciphertext bytes, and to let the attacker deliver ciphertext to a decryption oracle in a byte-by-byte fashion. They then proved that their new definition of SSH-CTR is secure in the new model. Because this new analysis more accurately captures how SSH is defined in the RFCs and how it's implemented, we can be more confident that this provable security result is more meaningful in practice.

The Future

Provable security is maturing into a very useful tool that can and should play an important role in the design and analysis of systems using cryptography. However, we've seen that protocols can be vulnerable to attacks outside the scope of the existing formal analyses. Nonetheless, such analyses can rule out large classes of attack. This leaves us with two major questions: How can we make the formal analyses more closely related to real implementations? And how can we better integrate formal analyses with the design process?

Making Theory More Applicable

In the SSH example, we saw that it's possible to perform analyses that accurately capture how schemes

Provable security is maturing into a very useful tool that can and should play an important role in the design and analysis of systems using cryptography.

operate in practice. Researchers are actively pursuing further work in this area, including the examination of how hardware and software side-channel attacks can be captured by formal analyses. For example, leakage-resilient cryptography is a particularly active research area that attempts to reason about security when the attacker has access to some secret information gained using a side channel.

When considering how to make the theory more practical, we still have at least one fundamental question to answer: How do we know exactly which protocol features are critical to its security and therefore must be included in the formal analysis? Many fine-grained details in cryptographic implementations might ultimately play a significant role in the protocol's overall security. As we've shown, omitting one of these details from a formal analysis can have a large effect on how that analysis applies in practice. On the other hand, Paterson and Watson's analysis of SSH attempts to model exactly how the RFCs specify the SSH BPP and how it's implemented in OpenSSH. Because of the model's detail and accuracy, we now have more confidence that no significant attack vectors were omitted. That said, Paterson and Watson's approach involves manual code inspection and proof generation. In the longer term, we hope that developers will create automated tools to assist in this kind of task. However, even this might not be sufficient-their analysis doesn't consider any issues arising from compression, which can play an important role in determining protocol security.⁷

Phillip Rogaway and Till Stegers have taken an alternative to Paterson and Watson's approach.¹⁵ They consider a protocol to consist of two parts: a partially specified protocol (PSP) and the protocol details (PD). The PSP will be the protocol's cryptographic core and is strictly defined by the protocol's specifications documents. The PD encompasses additional features necessary for an implementation but left open for the implementer to choose. Again, these additional choices sometimes introduce security vulnerabilities. To address this, Rogaway and Stegers give the adversary control over the PD, then attempt to prove the PSP's security. The idea is that if we can prove the PSP is secure when the adversary has control over the PD, then the full protocol (PSP, PD) must be secure in practice for any PD implementation. This is an intriguing approach that we expect to see further developed.

Bringing Theory into Practice

Today, formal protocol analyses are predominantly performed in a reactive manner: typically, someone designs and specifies a protocol, and only later does someone perform a formal analysis. Along with this approach, we've seen the rise of a model-attackremodel cycle, with the models gradually being upgraded to more closely reflect practice in light of new attacks. It's reminiscent of cryptography's break-fix cycle but has the advantage that, at each iteration, the security proof rules out a larger class of attacks. In an ideal world, we'd perform proactive formal analyses as part of the design process, before systems become widely deployed and therefore harder to change. By bringing together analysis and design, we can get a better idea of how various options might affect system security. Specification documents should include formal analysis results, giving the implementer a clear rationale for choices made in the design phase. However, this requires careful presentation, with provable security results presented in a simple and understandable manner for the consumption of practitioners.

n our view, better communication is key to uniting the theoretical and applied communities in cryptography. Without this, the two communities will continue to pull in opposite directions. By engaging both theoreticians and practitioners in a common endeavor—that of improving the security of real systems—we will foster a greater understanding of what provable security results actually mean and speed their adoption by practitioners. We hope that this article provides a small step in this direction. □

Acknowledgments

Vodafone Group Services Limited, a Thomas Holloway Research Studentship, and the Strategic Educational Pathways Scholarship Scheme (Malta), partly financed by the European Union–European Social Fund support Degabriele's research. An EPSRC Leadership Fellowship, EP/ H005455/1, supports Paterson's research. An EPSRC Industrial CASE studentship sponsored by BT Research Laboratories supports Watson's research.

References

- N. Koblitz and A. Menezes, "Another Look at 'Provable Security," J. Cryptology, vol. 20, no. 1, 2007, pp. 3–37.
- C. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical J.*, vol. 28, no. 4, 1949, pp. 656–715.
- S. Goldwasser and S. Micali, "Probabilistic Encryption," J. Computer Systems Science, vol. 28, no. 2, 1984, pp. 270–299.
- S. Micali, C. Rackoff, and B. Sloan, "The Notion of Security for Probabilistic Cryptosystems," *CRYPTO* 1986, LNCS 263, Springer, 1986, pp. 381–392.
- M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," *ASIACRYPT 2000*, LNCS 1976, Springer, 2000, pp. 531–545.
- H. Krawczyk, "The Order of Encryption and Authentication for Protecting Communications (or How Secure Is SSL?)," *CRYPTO 2001*, LNCS 2139, Springer, 2001, pp. 310–331.

- 7. T. Kohno, "Attacking and Repairing the WinZip Encryption Scheme," ACM Conf. Computer and Comm. Security, ACM Press, 2004, pp. 72-81.
- 8. B. Canvel et al., "Password Interception in a SSL/TLS Channel," CRYPTO 2003, LNCS 2729, Springer, 2003, pp. 583-599.
- 9. K.G. Paterson and G.J. Watson, "Immunising CBC Mode against Padding Oracle Attacks: A Formal Security Treatment, SCN, LNCS 5229, Springer, 2008, pp. 340-357.
- 10. J.P. Degabriele and K.G. Paterson, "On the (In)security of IPsec in MAC-then-Encrypt Configurations," Proc. 17th ACM Conf. Computer and Comm. Security (CCS 10), ACM Press, 2010, pp. 493-504.
- 11. M. Bellare, T. Kohno, and C. Namprempre, "Breaking and Provably Repairing the SSH Authenticated Encryption Scheme: A Case Study of the Encodethen-Encrypt-and-MAC Paradigm," ACM Trans. Information and Systems Security, vol. 7, no. 2, 2004, pp. 206-241.
- 12. W. Dai, "An Attack Against SSH2 Protocol," 6 Feb. 2002; www.ietf.org/mail-archive/text/secsh/2002-02. mail.
- 13. M.R. Albrecht, K.G. Paterson, and G.J. Watson, "Plaintext Recovery Attacks against SSH," IEEE Symp. Security and Privacy, IEEE CS Press, 2009, pp. 16-26.
- 14. K.G. Paterson and G.J. Watson, "Plaintext-Dependent Decryption: A Formal Security Treatment of SSH-

Publishing Se

- CTR," EUROCRYPT 2010, LNCS 6110, Springer, 2010, pp. 345-361.
- 15. P. Rogaway and T. Stegers, "Authentication without Elision: Partially Specified Protocols, Associated Data, and Cryptographic Models Described by Code," 22nd Computer Security Foundations Symp. (CSF 09), IEEE CS Press, 2009, pp. 26-39.

Jean Paul Degabriele is a PhD student at Royal Holloway, University of London. His research interests include cryptography and network security. Degabriele has an MSc in information security from the University of London. Contact him at j.p.degabriele@rhul.ac.uk.

Kenneth G. Paterson is a professor of information security at Royal Holloway, University of London. His research interests include cryptography and information security. Paterson has a PhD in mathematics from the University of London. He's a fellow of the IMA and a Journal of Cryptology editorial board member. Contact him at kenny.paterson@rhul.ac.uk.

Gaven J. Watson is a postdoc at the University of Calgary. His research interests are in cryptography and network security. Watson has a PhD in mathematics from Royal Holloway, University of London. Contact him at gavenjwatson@gmail.com.

handles the details

so you don't have to!

Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.

Professional management and production of your publication

- Inclusion into the IEEE Xplore and CSDL Digital Libraries
- Access to CPS Online: Our Online Collaborative Publishing System
- Choose the product media type that works for your conference: Books, CDs/DVDs, USB Flash Drives, SD Cards, and Web-only delivery!

Contact CPS for a Quote Today!

www.computer.org/cps or cps@computer.org



IEEE (computer society