Integrating Historical Security Jewels in Information Assurance Education

Jeffrey T. McDonald and Todd R. Andel | University of South Alabama

Secure communication and protocol analysis are key facets of security education, and previous research provides some foundational tenets thereof.

For better or worse, the world we live in relies on cybersystems for government, business, and industry. Securing data, communications, and applications in cyberspace represents an ever-increasing challenge. Although new technologies and threats evolve side by side, the basic principles of security remain stable. As we train and educate the information assurance (IA) security workforce, we observe that certain pieces of security research are foundational and timeless—they are jewels that deserve specific attention and focus.

One of the most widely accepted computer security principles existed well before modern computers. Auguste Kerckhoffs, a French military officer, published essays on military cryptography in the *Journal of Military Science* in 1883 that described the design principles for a military cipher system.[1] His timeless advice stated that a system should be secure, even if the enemy has a copy of it to examine. Claude Shannon reiterated this in modern terms,[2] and it has become the guiding principle for modern cryptographers and open source software providers.

In this article, we review jewels that remain foundational in terms of IA education, which help us formulate key learning objectives for future security professionals (see Table 1). These jewels will help future professionals take a systems-level view of security for how cryptography is used and implemented. They also serve as validation points to measure IA curricula.

Kerckhoffs and Secure System Design

In laying the foundations for future IA professionals, one of the key areas to be addressed is how to design secure systems for transmitting sensitive data. Aside from the knowledge of how cryptographic protocols work (which is covered in most data security and cryptography courses), the way in which systems create confidentiality is just as important. It is vital to understand how computationally hard problems bolster cryptographic primitives. Kerckhoffs described an ideal for how secure systems should be built,[1] and this principle continues to stand the test of time. As general goals, Kerckhoffs recommended cryptosystems that

- are easy to use,
- are portable,
- adapt to different media,
- are practically indecipherable (apart from the key),
- have keys that are communicable and changeable, and
- don't rely on secrecy for their designs.

Systems created using these tenets will likely retain a long-term strategic advantage over adversarial attacks.

 Copublished by the IEEE Computer and Reliability Societies

Table 1. Jewels of security education and the objectives they teach.

Researchers	Security objectives	Disciplines	Applications
Auguste Kerckhoffs[1]	Cryptographic systems' security shouldn't rely on obscurity for their secrecy.	Mathematical foundations, cryptographic algorithm design, and cryptographic primitives	Factoring, elliptic curves, quantum states, public key, private key, encryption, decryption, and signatures
Whitfield Diffie and Martin Hellman[6]	Mathematical primitives can help achieve privacy and authentication.		
Ronald Rivest, Adi Shamir, and Len Adleman[8]			
Dan Boneh, Richard DeMillo, and Richard Lipton[11]	Algorithms that are provably secure in theory might still have unexpected vulnerabilities when implemented in real systems.	Implementation and realization	Side-channel analysis and reverse engineering
Paul Kocher, Joshua Jaffe, and Benjamin Jun[12]			
Roger Needham and Michael Schroeder[7]	Mathematical primitives can help achieve authentication and secure message passing.	Communication protocols	Key exchange, message exchange, and identity verification
Danny Dolev and Andrew Yao[9]	Expect the limits of cryptographic security to be tested by polynomially bounded adversaries.		
Gavin Lowe[10]	Using cryptographic primitives doesn't guarantee privacy or authentication.		

Note that in such an approach, the only requirement for secrecy is in the key material itself.

Many in the modern cryptographic community have adopted the process of creating open and public algorithms. Alternatively, systems that try to keep algorithms hidden from the public are typically put in the category of "security by obscurity" and shunned by the cryptographic community. In building IA curricula, we stress that keeping the internals of cryptographic protocols secret doesn't add to the security of the system as a whole. First and foremost, we assume, as Shannon observed, that "the enemy knows the system."[2] Moreover, we assume that even if the enemy doesn't currently know the system, he or she eventually will. It's a harsh reality that many systems fall into adversarial hands for unintended reasons, often with severe consequences. Secret algorithms therefore pose risk to the overall system because multiple versions of algorithms, keys, and system components must be maintained. If the algorithm or a particular implementation is kept secret, it must be handled in the same way as the key, which will hinder the system's ease of use and portability.

Although Kerckhoffs's principle doesn't specify that you have to publish a cipher algorithm for public review, it does suggest that publication and review (even when visible to enemies) will not hurt but rather improve security. For new IA professionals, this revelation seems counterintuitive, but the practice has worked in recent history to produce best-of-breed competitions, such as the Advanced Encryption Standard (AES) that adopted the Rijndael protocol. In this case, the US National Institute of Standards and Technology worked with industry and the cryptographic community to develop an information-processing standard that would protect government and private sector information for several decades. The public vetting process only made the final standard stronger.

History has shown that cryptosystems are vetted best through continual feedback from professional cryptographers. For example, the US National Security Agency originally invented the Skipjack cipher as a classified protocol that was used in the proprietary Clipper chip tamper-proof telecommunications. When Skipjack was made available for public review, cryptographers discovered attacks on all of its 32-round traditional Feistel network structure.[3] Matt Blaze later reported the existence of a vulnerability in Clipper that allowed recovery of its embedded encryption key.[4] This is a classic example of how peer review could have strengthened the originally classified algorithm's security and its associated implementation in real hardware.

Future IA professionals should also understand that Kerckhoffs's maxims are only one aspect of secure system development. The principles primarily concern cryptosystems and don't always generalize to other aspects or types of system development. There's nothing inherently wrong with keeping system details secret, especially when there's a limited community of qualified reviewers or when the advantages gained by adversaries outweigh the benefits of openness. As Bruce Schneier

1:	A → AS	:	A, B
2:	AS → A	:	$\{PK_B, B\}\ SK_{AS}$
3:	A → B	:	$[\mathcal{N}_a, A]PK_B$
4:	B → AS	:	B, A
5:	AS → B	:	$\{PK_A, A\}\ SK_{AS}$
6:	B → A	:	$[\mathcal{N}_a, \mathcal{N}_b]PK_A$
7:	A → B	:	$[N_b]PK_B$

Figure 1. The Needham-Schroeder public-key protocol, which aims to provide authentication between two nodes without a priori security association.

points out, the best practice is to minimize the number of secrets and only use obscurity when there's a valid reason, but not to rest security entirely upon it.[5]

Fundamentals of Security Protocol Design

Cryptographic primitives are basic operations produced by using symmetric or asymmetric algorithms to encrypt, decrypt, and sign messages. Communication protocols are sequences of messages passed between parties that use primitives to enforce privacy, authentication, and possibly nonrepudation. As another key learning outcome in IA curriculums, future professionals should understand that the mere existence of provably secure ciphers with associated primitives doesn't guarantee that the primitives will be used correctly to provide security in a communication protocol. In other words, you can have a provably secure cryptographic protocol, but still use it the wrong way. The manner in which ciphers are applied to sending and receiving messages is the study of communications protocol design. Channels over which we communicate are assumed to be inherently insecure, and even those that have access control policies (such as Department of Defense networks) require privacy for message exchanges between parties. Protocols define the rules for beginning and ending a message transfer, authenticating users on each side of the protocol, and detecting errors.

We can trace public-key cryptography's introduction back to 1976 with work done by Whitfield Diffie and Martin Hellman.[6] Their intent was to minimize the need for a priori distribution of shared secret keys and to provide capability for digital signatures for messages. As they saw it, the problem with shared secret key systems was the need to have a unique key for each potential connection, which would then require each key to be computed and shared before use. This approach is unrealistic for a large number of users n, which would require $(n^2 - n)/2$ unique keys. Diffie and Hellman's approach was to provide each node with a key pair—E for enciphering and D for deciphering. In this scheme, key E can be publicly known for any other node to create encrypted messages that only the holder of the matching key D can decipher. Using this scheme in a reverse order also allows the key holder of D to sign messages (although it's not recommended that the same key pair be used for both purposes).

Diffie-Hellman security isn't based on the secrecy of their process but instead on the infeasibility of computing a node's secret key E from the corresponding public key D. This introduces the concept of computational security for IA education. Our goal in teaching computational security is to instill the concept that we must develop security protocols that allow for efficient computation for the intended user but that are still secure from a polynomial time–bounded attacker. Although no protocol, other than using a one-time-pad for a key lookup and substitution, is unconditionally secure against an unbounded and unlimited attacker, protocol designers must focus on security in the bounds of computational feasibility.

Needham-Schroeder Public-Key Protocol

Roger Needham and Michael Schroeder introduced their public-key protocol in 1978 as a means to provide authentication service between two nodes, represented as Alice (A) and Bob (B).[7] As Figure 1 shows, the protocol relies on public-key cryptography and an authentication server (AS). The format of the message indicates sender → receiver : message. AS acts as a key repository, storing the public key, whereas PK_X represents node X's public key for each node in the system. Each node in the system must store its own private or secret key as SK_X and a single public key for the key server as PK_{AS}.

The Needham-Schroeder public-key protocol provides numerous IA education concepts. First, the protocol allows two nodes to set up an authenticated channel. Once the connection is established, the nodes can use each other's public keys for private communication or, more typically, use the random nonces to generate private session keys for use with faster, traditionally shared secret key block ciphers such as the Triple Data Encryption Standard or AES algorithms. In addition to the primary concepts, there are also other significant IA concepts to consider.

Random nonces. The concept of random one-time values (N_a and N_b) can be used to establish temporary shared session keys. It can also help distinguish between true randomness and pseudorandomness.

Trust and authentication. This entire scheme is based on the trust in the authentication server and on the premise that the public key used for the authentication server is valid.

Secure message passing. We can learn from the concepts of both encryption (denoted as $[]PK_X$) and signatures

(denoted as $\{\}SK_X$), noting the difference in what material can remain public (but authenticated through signatures) versus what material must remain secret.

RSA Public-Key Protocol

Whereas Needham and Schroeder illustrated a secure message exchange that can use any public-key protocol, the Rivest-Shamir-Adleman (RSA) public-key protocol illustrates the mathematical properties that make a protocol computationally secure to polynomial time–bounded attackers. IA professionals, at a minimum, should understand the profound impact of the RSA algorithm, introduced in 1978.[8] It's a mathematical method of encryption based on modular exponentiation. In its basic operation, two large prime numbers (p and q) are multiplied to produce a modulus n. Euler's totient $\phi(n)$ then produces an encryption (private) and decryption (public) key pair. The encryption key E is chosen as an integer such that $1 < E < \phi(n)$. The decryption key D is chosen as the multiplicative inverse of E, where $D = E^{-1} \bmod \phi(n)$. Although the protocol is fully open (following Kerckhoffs's principle), the sole security is based on the difficultly for an attacker to factor the public modulus n to determine the undisclosed factors p and q.

Fundamentals of Protocol Analysis

In addition to illustrating security protocol design and structure for teaching IA concepts, protocol analysis provides a critical counterpart to its own IA concepts. An important concept in security protocol analysis is the attack model Danny Dolev and Andrew Yao introduced in 1983.[9] The Dolev-Yao model follows Kerckhoffs's principle in that the attacker can perform any operation other than breaking the protocol's underlying cryptographic primitives.

The attacker is considered a valid node or trusted insider of the system; thus, he or she can obtain original information on his or her own public- and private-key pair and the public keys of all other nodes in the system. The attacker can initiate or respond to any connection request. An attacker is assumed to have the ability to capture any message in the system and can influence the messages it observes via replaying, modifying, or dropping messages. All these actions can occur without breaking the underlying cryptographic primitives. The attacker's overall goal is to obtain secret information or to trick a node into authentication on behalf of another node.

Without the use of the Dolev-Yao model, analysis techniques might inappropriately trust nodes or limit intruder capabilities with improper assumptions. Dolev-Yao also illustrates to future IA professionals that the use of cryptography in and of itself doesn't guarantee privacy or authentication.

1:	$A \rightarrow AS$	:	A, B
3:	$A \rightarrow B$	:	$[\mathcal{N}_a, A]PK_B$
6:	$B \rightarrow A$	:	$[\mathcal{N}_a, \mathcal{N}_b]PK_A$
7:	$A \rightarrow B$	:	$[N_b]PK_B$

Figure 2. Lowe's simplified Needham-Schroeder public-key protocol, which provides the same authentication as the original protocol but assumes each node already possesses the required public keys.

Analyzing Needham-Schroeder

Flaws in security protocols can be subtle. The Needham-Schroeder public-key protocol had a somewhat obvious flaw that went undiscovered until Gavin Lowe published his attack nearly 17 years later.[9] This discovery illustrates that even generally accepted protocols can contain vulnerabilities and focuses the IA professional on rigorous analysis techniques. To attack the protocol, Lowe removed the unneeded complexities from Needham-Schroeder, leaving a simplified version (see Figure 2). Lowe based this simplification on the assumption that each node already possesses the required public keys because the signature operation of the key server is considered unforgeable.

Using this simplification, Lowe was able to discover the attack illustrated in Figure 3. The Lowe attack shows that a third authorized network user (denoted as inside intruder I) can perform a man-in-the-middle attack, convincing a node that it has authenticated a session with a different user. As node A attempts to authenticate a session with node I, I decrypts the values of A's request with its private key and forwards these values under B's public key to node B, with the intent of creating a spoofed session with B. Basically, I(A) depicts I spoofing that A has sent the message. As a result, B believes it has opened an authenticated session with A, but the session is actually with node I. The vulnerability exists because when B thinks it's responding to A (that is, $B \rightarrow A : \{N_a, N_b\}K_a$), I simply relays this information to A. I uses node A as an oracle to obtain B's nonce, which was originally encrypted with A's public key. The significance of the attack is now that I—even though an authorized insider—can interact with node B and receive information that might have been private to only A.

Fortunately, Lowe provided a simple correction to the protocol. Lowe's fix forces B to include a reference to itself in the encrypted response: $B \rightarrow A : \{B, N_a, N_b\}K_a$. This step ensures that if I relays this packet to A, A will know that the packet originated from B rather than I. Lowe's fix shows the significance of providing proper identification within encrypted messages.

Lowe's attack illustrates the complexity of using cryptographic primitives correctly; subtle flaws can

$$
\begin{array}{lcl}
A \rightarrow I & : & \{\mathcal{N}_a, A\}\mathcal{K}_i \\
I(A) \rightarrow B & : & \{\mathcal{N}_a, A\}\mathcal{K}_b \\
B \rightarrow I(A) & : & \{\mathcal{N}_a, \mathcal{N}_b\}\mathcal{K}_a \\
I \rightarrow A & : & \{\mathcal{N}_a, \mathcal{N}_b\}\mathcal{K}_a \\
A \rightarrow I & : & \{\mathcal{N}_b\}\mathcal{K}_i \\
I(A) \rightarrow B & : & \{\mathcal{N}_b\}\mathcal{K}_b
\end{array}
$$

Figure 3. Lowe's Needham-Schroeder public-key protocol attack. The attacker I can set up an authenticated channel with node B while it appears to node B that the secure channel is set up with node A.

exist in relatively simple protocols. We can use Lowe's protocol simplification step to introduce the concept of abstraction in IA education: to find flaws, it's convenient to simplify a protocol for analysis purposes. Abstraction lets us remove portions of a security protocol while ensuring that the core protocol semantics aren't altered.

Analyzing RSA

The advent of available and affordable embedded hardware solutions over the past decade has given a false sense of security in certain cases. IA education should address these limitations and make sure that future professionals are aware of the limitations for hardware realizations. The fact that many algorithms can be directly implemented at the hardware level—apart from the traditional, general-purpose, processor-based software model—has opened up greater avenues of efficiency and security. However, the actual way in which cryptographic ciphers are implemented can be vulnerable to exploits. Dan Boneh and colleagues highlighted the ability to break security protocols without being able to directly attack a modulus or enumerating its key space; they showed that hardware implementations can be attacked directly through transient, latent, or induced faults.[11]

Paul Kocher and colleagues did groundbreaking work on side-channel analysis (SCA) attacks against RSA.[12] Although the RSA algorithm's plaintext-to-ciphertext message stream is provably secure, hardware implementations are still vulnerable to SCA attacks.

Assume an RSA algorithm is burned into a physical circuit for efficiency purposes, and then this hardware is subsequently lost or stolen. If an adversary can determine this device's private key, then that adversary can view all messages that are or have previously been sent to this device. Typical RSA implementations process the modular exponentiation in iterations; they look at an individual key-bit per iteration and

- perform a squaring operation if the current key-bit is a 0 or
- perform both squaring and multiplying operations if the current key-bit is a 1.

In hardware, these two operations generate different switching activity in the transistors and thus produce different power levels in the hardware. Using a noninvasive electromagnetic (EM) probe, an attacker can view the related EM power generated during these actions. Using this procedure, extracting the key from a hardware RSA implementation is relatively straightforward, as Figure 4 shows, where the high peaks indicate a key-bit as a 1 (the algorithm is executing a square and multiply operation) and low peaks indicate a key-bit as a 0 (the algorithm is performing a square operation).

From an IA education perspective, this attack shows the added need of securing implementations rather than just assuming that using a secure protocol algorithm ensures overall system security. SCA attacks on hardware have introduced an entirely new research area for IA professionals in developing SCA countermeasures for hardware implementations.

In preparing curriculum to train the future IA workforce, several key facets of security exist independent of the current state of technological advancement and adversarial attacks. Several of these jewels can serve as signposts to help frame our understanding of other security-related material. In the long run, curricula that incorporate these jewels will better prepare future IA professionals to defend the national cyberinfrastructure. ■

References

1. A. Kerckhoffs, "La Cryptographie Militaire," *J. Sciences Militaires*, Jan. 1883, pp. 5–83.
2. C.E. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical J.*, Oct. 1949, pp. 656–715.
3. E. Biham, A. Biryukov, and A. Shamir, "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials," *Proc. 17th Int'l Conf. Theory and Application of Cryptographic Techniques* (EUROCRYPT 99), Springer, 1999, pp. 12–23.
4. M. Blaze, "Protocol Failure in the Escrowed Encryption Standard," *Proc. 2nd ACM Conf. Computer and Comm. Security* (CCS 94), ACM, 1994, pp. 59–67.
5. B. Schneier, "Secrecy, Security, and Obscurity," *Cryptogram Newsletter*, Counterpane Internet Security, 15 May 2002; www.schneier.com/crypto-gram-0205.html.
6. W. Diffie and M. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, Nov. 1976, pp. 644–654.
7. R. Needham and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Comm. ACM*, Dec. 1978, pp. 993–999.

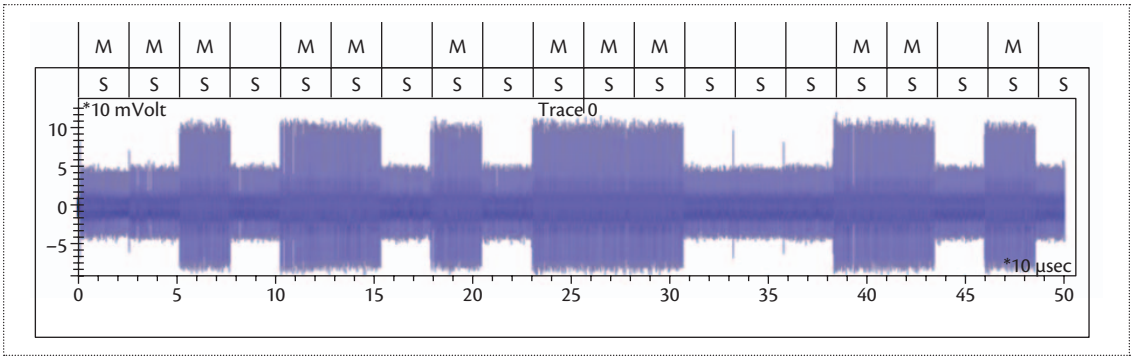

Figure 4. RSA side-channel analysis attack. Using this procedure, extracting the key from a hardware RSA implementation is relatively straightforward. The high peaks indicate a key-bit as a 1 and low peaks indicate a 0.

8. R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM*, Feb. 1978, pp. 120–126.
9. D. Dolev and A. Yao, "On the Security of Public Key Protocols," *IEEE Trans. Information Theory*, Mar. 1983, pp. 198–208.
10. G. Lowe, "An Attack on the Needham-Schroeder Public-Key Authentication Protocol," *Information Processing Letters*, Aug. 1995, pp. 131–133.
11. D. Boneh, R.A. DeMillo, and R.J. Lipton, "On the Importance of Checking Cryptographic Protocols for Faults," *J. Cryptology*, vol. 14, no. 2, 2001, pp. 101–119.
12. P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Proc. Advances in Cryptology* (CRYPTO 99), LNCS 1666, Springer, 1999, pp. 388–397.

Jeffrey T. McDonald is an associate professor at the University of South Alabama's School of Computing. His research interests include program protection and exploitation, secure software engineering, and information assurance. McDonald received a PhD in computer science from the Florida State University. He's an associate member of IEEE. Contact him at jtmcdonald@southalabama.edu.

Todd R. Andel is an associate professor at the University of South Alabama's School of Computing. His research interests include network security protocols, secure electronic voting, and information assurance. Andel received a PhD in computer science from the Florida State University. He's a member of IEEE. Contact him at tandel@southalabama.edu.