IEEE Technology and Society MAGAZINE

A PUBLICATION OF THE IEEE SOCIETY ON SOCIAL IMPLICATIONS OF TECHNOLOGY
VOLUME 33 | NUMBER 3 | FALL 2014

Technology for Collective Action

IEEE

COMMENTARY

ALAN G.
KONHEIM

To Let Them Monitor or Not ... Perhaps that is *Not* the Real Question

The United States is now debating the wisdom of limits on monitoring by the National Security Agency (NSA). While the privacy/security tradeoffs for even passive *monitoring* are complicated, articles on Google have suggested that the NSA's activities are more intrusive than previously thought. There is increasing evidence that NSA may be interfering with secured Internet transactions by using a backdoor attack [1],[1] "a method of bypassing normal authentication, securing illegal remote access to a computer, obtaining access to plaintext, and so on, while attempting to remain undetected." For example, the U.S. NSA appears to have been connected to a recent backdoor attack involving the random number generator (RNG) used in virtually every Internet transaction. This is reminiscent of the celebrated backdoor believed to have occurred in 1941 when the FBI "borrowed" codebooks from the Japanese consulate in Manhattan. I will explain in this article the role cryptography plays in our everyday lives, the current involvement of the NSA, and suggest a possible intrusion by them on our privacy which is hinted at in articles posted on Google.

After completing graduate school in 1960, I became a Research Staff Member at the IBM Thomas J. Watson Research Center in Yorktown Heights, NY. Six years later, the IBM Corporation entered the cryptographic business when Lloyd's Banking Group contracted with IBM to design a remote-terminal-oriented banking system. It led to the now ubiquitous Automated Teller Machines (ATM) at which cash dispensing and related banking transactions could be carried out. Until the application of encryption in ATM transactions, the only kid on the block with cryptographic competence was the NSA.

The business advantages of ATM-oriented banking were obvious to the banking community. It could be 1) available 24/7, 2) implemented outside of the banks at point-of-sales terminals in hotels or supermarkets, and 3) ATMs would not unionize nor require medical or retirement benefits. ATM banking was profitable for the banks and simplified cash management of vacations.

While soda and cigarettes had long been dispensed by machines, banks were uncomfortable with dispensing cash. The banks insisted the customer be authenticated before an ATM transaction proceeded further. To address the bank's concerns, IBM proposed ATM-authentication employing two cryptographically related customer identifiers. A customer would begin by inserting the bank card into the ATM terminal which would read the first customer identifier, the Personal Account Number (PAN). Next, the customer enters at a keyboard the second identifier, the Personal Identification Number (PIN). Cryptography provided the proper relationship between the PAN and PIN which was verified to enable a transaction. Banks warned customers to keep their PINs secret, and *not* to write them on their bank card.

In 1970, I managed a group in the Mathematical Sciences Department at the IBM Yorktown Research Center. Its task was to evaluate the appropriate encryption coding. Cryptography would become the focus of my professional life for over forty years.

Cryptography uses an algorithm to encode the original plaintext data into ciphertext, with the goal of completely disguising the original message. To achieve secrecy, every cryptographic algorithm requires a key. Until 1978, all cryptographic coding schemes were symmetric; if a plaintext message is encrypted with key K, the same key suffices to decipher the ciphertext and recover the original data. The number of possible

[1] The term *trap door* in [1, section II] is a synonym of *backdoor.*

Digital Object Identifier 10.1109/MTS.2014.2345171
Date of publication: 17 September 2014

keys must be large enough to foil key-trial, in which an intruder tests all possible keys, deciphering the ciphertext with each until it finds meaningful plaintext. Since the cryptographic algorithm is not secret, as in the widely used Data Encryption Standard (DES) and Advanced Encryption Standard (AES), the success in hiding the plaintext depends on keeping the key secret.

IBM Research was fortunate having hired Horst Feistel who had fled Germany in 1935. He was educated in Zurich and later worked at the Air Force Cambridge Research Center in Massachusetts where he learned the principles of cryptography. When he came to IBM, Horst used this knowledge to invent the cryptographic algorithm *Lucifer,* which morphed into the DES approved in 1977 as the Federal Information Processing Standard (FIPS) 46-1. While DES was reaffirmed as a standard four times, the National Institute of Standards (NIST) wrote in December 1998.

A public key cryptographic system assumes that with knowledge of the public key, it remains computationally infeasible to calculate the private key.

> At the next review (1998), the algorithm specified in this standard will be over twenty years old. NIST will consider alternatives which offer a higher level of security. One of these alternatives may be proposed as a replacement standard at the 1998 review.

In January 1997, NIST solicited proposals in the Federal Register for the AES. The cryptographic algorithm *Rijndael*, designed by the Belgian researchers Joan Daemen and Vincent Rijmen, was announced as the winning algorithm in October 2000 and is specified in FIPS-197. AES is structurally similar to DES but with a much larger nominal key size.

Cryptography has become both relevant and important in our lives, not only as a result of ATM-simplified banking, but due to the emergence of e-Commerce, the purchasing of products and services conducted over electronic systems like the Internet. In normal commercial transactions, the parties meet together and sign a contract specifying the details of their transaction. When something is purchased on the Internet, the buyer and the seller are not in physical contact and each party requires something from the other to be more certain of the contract. When a buyer makes a credit purchase on the Internet, the seller must receive authorization to charge the buyer's credit card. This authorization cites the purchase details and includes the buyer's credit card number, its expiration date and the card verification value.

These last pieces of information are sensitive and must be otherwise kept secret. How can a buyer securely transmit this information to the seller over the Internet? By enciphering the purchase authorization prior to transmission to the seller, it might prevent an Internet eavesdropper from recovering the credit card data. Since both DES and AES are symmetric cryptographic encryption, the same key must be known by both the seller and buyer. When the State Department needs to securely send a message from Washington to Moscow, it makes prior arrangements to have a courier deliver the key to the embassy in Moscow. This is not feasible in the environment of the Internet. How can the same secret key be made available to the two parties in e-Commerce? A new idea was needed.

A solution was proposed in a 1976 seminal paper [2] by Whit Diffie, then a graduate student at Stanford University and his advisor, Professor Martin Hellman, a former IBM Research Staff Member. They suggested the use of asymmetric encryption or a public key cryptographic system (PKS) which would use a non-secret public key and a secret private key. A PKS assumes that with knowledge of the public key, it remains computationally infeasible to calculate the private key. Because *a* symmetric algorithms are usually more computationally intensive than symmetric counterparts, the parties in e-Commerce would use a symmetric algorithm to encipher their communications. The buyer would 1) generate a traditional AES/DES key K, 2) PKS-encipher K with the seller's public key, and 3) transmit the resulting (enciphered) key to the seller. The seller would use its private key to PKS-decipher the encrypted key, thus discovering which K the buyer had selected. Thereafter, the buyer could provide credit card information securely to the seller using DES/AES encryption with key K.

Netscape developed the Secure Sockets Layer (SSL) protocol, precursor to the current Transport Layer Security (TLS) protocol for use with its Navigator browser, the Google Chrome of yesteryear. In the language of TLS and SSL the buyer is a client, the seller is a server and these parties are connected using a web browser, for example, Google Chrome. TLS specifies the steps needed to support the secure exchange of information over the Internet, by using the browser software to carry out the steps with only minimal intervention by the buyer. Once a secure socket connection was established, e-Commerce could be carried out securely and business could flourish.

Perfect, except the Diffie and Hellman's paper did not truly provide a viable example of a PKS. Two years later, Ronald Rivest (of M.I.T.), Adi Shamir (of the Weizmann Institute in Israel), and Len Adelman (then at M.I.T. and later at USC) produced the first true PKS, referred to as RSA [3].

But there still remains a problem. Adelman's student Loren Kohnfelder noted in 1978 [4] that all public-key cryptosystems are vulnerable to *spoofing* attacks; *to spoof* is to "cause a deception or hoax." The buyer thinks the Internet has connected his/her computer to the website of xxxairline.com, but alas, the buyer's usually trusty machine has been spoofed. Konhfelder observed that the seller had to be authenticated to the buyer and proposed a solution. He suggested users "… must place his enciphering algorithm (his public key) in the public file." These entries in the public file would establish the relationship between the identity of the seller xxxairline.com and the seller's public key. Konfelder referred to this proof of identity as a (public-key) "certificate." While the seller might require the buyer to also have a certificate in both SSL and TSL, the seller generally does not. Having received the authorization and credit card details from the buyer, the seller contacts the buyer's credit card issuer through a payment gateway, analogous to the familiar card-swiping devices in a supermarket, and the transaction is either accepted or declined.

Once a secure socket connection was established, e-Commerce could be carried out securely and business could flourish.

The International Telecommunication Union (ITU) Recommendation X.509 specifies how the certificate accomplishes authentication. X.509 supposes the existence of trusted Certification Authorities (CA) which would issue valid certificates. Examples of certificate issuers include digicert.com, verisign.com and geotrust.com. What does a certificate contain and how is it constructed?

To "hash" is to chop into small pieces. The website recipesource.com lists well over 100 recipes for the American quintessential dish corned-beef hash. My favorite is the famous Swiss maximum artery-clogger Berner Rösti combining grated potatoes, onion, garlic, and bacon. The X.509 certificate lists descriptors of the seller, including the seller's identity and public key. These descriptors are first hashed together and then enciphered using the CA's private key to derive the signature on the certificate. The CA's public key is made available to all browsers and can be used to check the signature and thus verify the public key of the purported seller.

To summarize, there are several steps in the SSL/TSL authentication process:

1) The buyer verifies the identity of the seller by using the seller's certificate to first authenticate the seller as the party to whom the web browser has connected to the buyer *and* next for the buyer to learn the seller's public key.
2) Once the seller has been authenticated, the buyer uses a random number generator (RNG) to select a "random" AES/DES session key K and transmits the enciphered version of K using the PKS employing the seller's public key. This session key K will only be used during this buyer-seller session.
3) The seller may determine K by deciphering the key-transmission ciphertext using the seller's private key.

Both parties now have the same key K and the deal can be struck. What could go wrong? Plenty!

- In 2005, X. Wang, Y. Yin and H. Yu proved that the hashing functions MD5 [5] and SHA-1 [6] used to make the certificates were defective. If they were used to hash, SSL/TLS might be circumvented implementing a rogue Certificate Authority [7] to steal a buyer's secret information.
- Perhaps, the RNG selected key is *not* randomly selected.
- Perhaps, the seller's or CA's private key is *not* so private.

These three exposures have serious and potentially harmful effects on privacy and security. While the Wang-Lin-Yu glitch might be fixed by changing the hashing method, the last two issues could be surreptitiously influenced by the NSA.

TLS/SSL uses different keys to secure an Internet transaction. For example:

- In Step 1, the public key of the CA is used to authenticate the certificate of the seller.
- In Step 2, a session key is randomly generated by the buyer to be used to securely exchange information during the TLS-session.
- In Step 2, the public key of the seller is used by the buyer to securely deliver the session key to the seller.
- In Step 3, the seller uses its secret key to obtain the session key selected by the buyer.

The exposure of these keys would have different harmful effects:

- If a CA's private key were revealed, it would allow the fabrication of bogus certificates compromising *all* sellers which used this CA to obtain a certificate.
- Even though the session key is chosen afresh during each session, a single compromise would reveal the credit card information of the buyer.
- If a seller's private key were revealed, it would compromise *all* TLS transactions with this server.

The NSA and its British counterpart the General Communications Headquarters (GCHQ) have enjoyed a distinguished history of success in cryptanalysis. The two decisive battles of World War II were the Battle of Midway (1942) in the Pacific and the Battle of the Atlantic (1939). The victory, which defeated the German U-boat attacks on U.S. shipping, was necessary in order for England to survive. These victories were achieved in a major way as a consequence of the cryptanalytic efforts of at NSA and GCHQ.

In the early part of the 20th century encryption employed electro-mechanical machines to perform polyalphabetic substitution of letters; the German Enigma, SZ40, and the Japanese PURPLE machines are three examples. The Enigma machine used rotors, the SZ40 used wired code-wheels, and the Japanese machines used telephone switches. Mechanical components limited the possible complexity of the encryption.

What could go wrong? Plenty!

The 1977 DES was different; it could be implemented as a program and there were no limits in principle to its complexity. The debate surrounding certifying DES as a standard focused largely to concerns about whether NSA intruded on its design. In fact, it didn't except for the choice of the strange key-length size of 56 bits. It is not possible to prove this was the only governmental interference, but I suggest that if they did more, then "how come no one has uncovered NSA's trap-door in the nearly 40 years?" AES may be an improvement on both DES and triple DES (DES3), but the question remains "are they really totally secure?"

After coming to UCSB, I spent seven summers at the NSA or IDA/CRD in Princeton which consults for NSA. I have great respect for many of the people I met. David Kahn describes Edgar Allen Poe's short story "The Gold Bug" [10]; in it, Poe wrote "yet it may be roundly asserted that human ingenuity cannot concoct a cipher that human ingenuity cannot resolve." I am no longer certain. Perhaps the Gershwin brothers made a more percipient judgment, when in *Porgy and Bess* they wrote the song *It Ain't Necessarily So*. Until a smart university cryptography guru finds a structural fault, 296 remains too big a number for key-trial, although quantum computers might give them a lift. If this be the case, what can NSA do to stay in the cryptanalysis business?

NIST Special Publication 800-90A [8] entitled "Recommendation for Random Number Generation Using Deterministic Random Bit Generators" describes several NIST approved random number generators used to generate a session key. If a random number generator had a structural weakness or introduced bias, this might permit an eavesdropper to determine the session key by key-trial, testing the *effectively* smaller set of keys.

The article appearing on the website *Arstechnica.com* entitled "How the NSA (May Have) Put a Backdoor in RSA's Cryptography: A Technical Primer" (http://arstechnica.com/security/2014/01/how-the-nsa-may-have-put-a-backdoor-in-rsas-cryptography-a-technical-primer/) describes the compromise of the session key when using one of the approved NIST 800-90A random number generators. The article asserts a deliberate weakness was introduced in TLS V1.2 as the result of a contract between RSA Incorporated and NSA. The claim is based on both the Snowden revelations and the 2007 work of Microsoft's Shumow and Ferguson explaining the flaws in the Dual_EC_DRBG's, one of the NIST 800-90A generators. To be fair, there are alternate generators and apparently RSA Security LLC, has recently disowned the offending RNG.

However, revealing the CA's private key is the more serious and effective intrusion and might be hard to prove. There are two possible scenarios:

1) General_NSA takes the CEO of jungle.com to lunch on K Street and asks him or her. "Help us protect the U.S. Reveal to us the private key used in signing jungle.com's certificate. We're the good guys and know how to keep a secret!" If the good girl/guy CEO agrees, then all of jungle.com's TSL transactions are made transparent. As the FBI learned in 1941, one peek is better than 10–20 Cray supercomputers.
2) General_NSA takes the CEO of the CA xxxsecurecert.com to lunch (or dinner) and makes the same argument asking for the CA's private key. If there is acquiescence, all of the sellers who certificates were issued by this CA would be vulnerable to attack.

Is this scenario far-fetched? Perhaps, but the Feb. 28, 2014, web article [9] "Lavabit's Ladar Levison on Snowden, Why He Shut Down, and How to Beat the NSA" states that "Levison was prohibited from discussing any details of the case until last October, when the court unsealed a portion of the documents. The unsealed records reveal that the FBI was demanding access to Lavabit's Secure Sockets Layer (SSL) keys, which would essentially allow the agency access to all messages on Lavabit's server. While the FBI was ostensibly targeting only a single user, Levison was unwilling to sacrifice the privacy of his other 400,000+ users."

If the FBI can do it, can **N**o-**S**uch-**A**gency be far behind?

The story of intrigues is not over, although the NSA fingerprints are less obvious. In April 2014, the *Heartbleed* bug was revealed. It results from a weakness in an *OpenSSL*, an implementation of SSL/TLS, the basis of communication security and privacy over the Internet. It affects the security/privacy of web transactions, email, and instant messaging. The Heartbleed bug allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software. The bug may compromise the secret keys used to identify the service providers and to encrypt their transmissions, the names and passwords of the users, and the actual message content. This allows attackers to eavesdrop on communications, steal data directly from the services and users, and to impersonate services and users.

We live in dangerous times and our security may be threatened by many sources. There are legitimate roles for NSA, but undermining the security of transactions over the Internet seems too high a price to pay. Perhaps, they should use the front door.

Author Information

Alan Konheim is Professor Emeritus in the Computer Science Department, University of California, Santa Barbara, CA 93101, U.S.A. Email: konheim@cs.ucsb.edu.

Acknowledgment

The author acknowledges with thanks the many helpful suggestions, comments, and encouragement made by Andrew Pickholtz.

References

[1] H.E. Petersen and R. Turn, "System implications of information privacy," in *Proc.AFIPS Spring Joint Computer Conf.* AFIPS: 1967, vol. 30, pp. 291–300.

[2] W. Diffie and M.E. Hellman, "New directions in cryptography," *IEEE Trans. Information Theory*, vol. IT-22, pp. 644–654, Nov. 1976.

[3] R.L. Rivest, A. Shamir, and L. Adelman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[4] L.M. Kohnfelder, "Towards a practical public-key cryptosystem," M.I.T., Cambridge, MA, Bachelor's thesis, May 1978.

[5] X. Wang and H. Yu, "How to break MD5 and other hash functions," in *Proc. Advances in Cryptology-Eurocrypt 05.* Springer, 2005, pp. 1–18.

[6] X. Wang, Y. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Proc.Advances in Cryptology-Crypto 05*.Springer, 2005, pp. 17–36.

[7] M. Stevens, A. Sotirov, B. Applebaum, A. Lenstra, D. Molinar, D.A. Osvik, and B. de Weger, "Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate," in *Proc. Crypto `09.* Springer-Verlag, 2009.

[8] National Institute of Standards, "Recommendation for random number generation using deterministic random bit generators," NIST Special Publication 800–90A, Mar. 10, 2009; updated Feb. 14, 2013.

[9] Z. Weissmueller, "Lavabit's Ladar Levison on Snowden, why he shut down, and how to beat the NSA," *reason.com*, Feb. 28, 2014; http://reason.com/reasontv/2014/02/28/lavabits-ladar-levison-on-snowden-why-he.

[10] Writing contest – winning entry, *Philadelphia Dollar Newspaper*, vol. 1, no. 23, pp. 1–4, June 28, 1843.