Strong Cryptography and Public Policy

a talk delivered to the Connecticut Bar Association,
May, 2002

by

Charles W. Neville, Ph.D.
cwneville@cwnresearch.com

Ladies and Gentlemen of the Connecticut Bar Association:

0. Introduction.

It is an honor to be invited to speak here. I want to make the case for two propositions:

PHILOSOPHY is worth supporting.
Even PURE MATHEMATICS is worth supporting.

I know this talk is supposed to be about cryptography, and we will certainly get to that. In fact, you may learn more about cryptography in the next 45 minutes than you ever thought you wanted to know. But first, we need to examine some history, in the spirit of George Santayana's famous dictum,

Those who fail to learn from history are condemned to repeat it.

1. How Computation Came To Be.

At the 1900 International Congress of Mathematicians, David Hilbert proposed a list of 23 problems. These set the agenda for much of the mathematical research of the 20th century, and some of them are still unsolved. Remarkably, one problem, the 2nd on the list, produced the following great advances, which many would consider hallmarks of 20th century science and thought:

Russell and Whitehead's reduction of arithmetic to symbolic logic.

Kurt Goedel's famous theorem that mathematics is fundamentally incomplete, that there are mathematical truths which we will never be able to prove or disprove.

Alan Turing's invention of the first (abstract) computer, the Turing machine.

The Church – Turing thesis that all sufficiently general forms of computation are equivalent, in the sense that anything computable in one is computable in all.

Burks, Goldstein and Von Neumann's invention of the stored program computer.

Surely this last is one of the most valuable technological advances ever, comparable in impact to the invention of printing, and perhaps even to the invention of the wheel. The others were essential steps along the road to Burks, Goldstein and Von Neumann's great invention. And all were direct consequences of attempts to solve Hilbert's 2nd problem. So let us consider the dramatis personae:

Russell and Whitehead were PHILOSOPHERS.

Hilbert, Goedel, Turing, Church, and Von Neumann were PURE MATHEMATICIANS.

Surely the invention of the computer is the greatest payout society has ever received for the support of PHILOSOPHY and PURE MATHEMATICS. Do you think our friends Bill and Steve should endow a chair of philosophy, and perhaps even a chair or two of pure mathematics?

2. The Relation To Cryptography And Other Things.

I know you all have a burning desire to learn about cryptography, but first, in the spirit of George Santayana's famous dictum, we need to consider other things.

Bertrand Russell wrote much of his and Whitehead's great work, reducing arithmetic to symbolic logic, while in JAIL in 1918 on war resistance charges. The moral is, treat war resistors and other prisoners well; you never know what contributions they may make.

Alan Turing directed the successful effort to break the German Enigma Codes during World War II. Some say he saved his country. After the war, he was jailed on homosexuality charges and committed suicide. The moral is, don't side with the moral majority.

It is unclear whether the methods Turing and his coworkers invented to break the German Enigma Codes were also used by Cmdr. Rochefort and Col. Friedman to break, respectively, the Japanese Naval Codes and the Japanese Diplomatic Codes. Cmdr. Rochefort broke regulations by giving Adm. Nimitz raw intelligence about the impending Japanese attack on Midway. Using this intelligence, Nimitz turned the tide in the Pacific by winning the Battle of Midway. After the war, Cmdr. Rochefort was denied promotion and forced into early retirement for disobeying regs. The moral is, that's why it's called "Naval Intelligence."

The official (until very recently) U.S. Government encryption standard, the Digital Encryption Standard or DES, is not based on the German Enigma system, but every cryptographer who worked on it was acutely aware of the fatal weaknesses of the supposedly unbreakable Enigma. While the DES can be cracked using a lab full of computers and several days of computation, breaking it is beyond the capacity of Harry Hacker or Script Kiddie. By contrast, large American companies such as Microsoft and Adobe have insisted on using proprietary encryption methods which are really easy to break. The joke is, "What's the difference between Microsoft and the Germans?" Answer, "Both make you do it their way, but with the Germans it works."

The list of consequences of the work surrounding Hilbert's 2nd problem AFTER Burks, Goldstein and Von Neumann's 1944 invention of the stored program computer includes most of contemporary computer science, excepting only circuit design. Highlights include:

1. Higher level computer languages such as FORTRAN, C and Java.

2. Virtual machines, including the emulation of the Intel instruction set on AMD's Athalon chip and the Java virtual machine. (Can you see how this follows from the Church – Turing thesis? For the curious, the answer is contained in appendix C.)

3. The entire field of Artificial Intelligence. (Can you see how this also follows from the Church – Turing thesis? Again, the answer is contained in appendix C.)

4. One way functions and the identification, by Cook, of certain maximally hard problems solvable by guesswork, such as the travelling salesman problem. (For the EXTREMELY CURIOUS, The travelling salesman problem is described in appendix D.) The problem of factoring large numbers, which underlies much of public key cryptography, is (thought to be) very hard and nearly as hard as any of the maximally hard problems.

5. Public key cryptography and the invention of (almost) provably unbreakable codes. Every time you use the Internet to buy something, you use a public key encryption method as part of the secure sockets layer.

For the cognoscenti, and for those who demand fairness to the field of COMPUTER SCIENCE as it emerged in the 1950's and beyond, it is important to note that one can't simply read these results off from work done on Hilbert's 2nd problem, except perhaps in retrospect. Those who actually produced these results did much work of the highest quality, and most of them would consider themselves to be computer scientists rather than mathematicians.

3. Public Key Cryptography.

Cryptography is the art and science of putting messages into code. The art and science of breaking codes is also part of cryptography.

An essential problem with codes is KEY DISTRIBUTION. To put a message into code, or to decode a message, you have to know not only the coding method, but also a key. Poor key distribution was one of the mistakes the Germans made in World War II with their Enigma Codes. They would send out today's key over the radio first thing in the morning using yesterday's key. This allowed the British, who had to work for a day or two to break the Enigma Code without the key, to read coded messages in real time. And just to make things even easier for the British, the Germans would send out a message using well known German swear words (really strong stuff like "Donner Wetter") so everyone could test their new settings. This way, when the British didn't have today's key because the Germans had been smart and had used a pre-prepared list of keys in a code book, the British could still break the code and start reading the key sequence again.

Public key cryptography solves the key distribution problem. Here's how it works.

On any given day, there are TWO keys,

A PUBLIC KEY which you make public, and which is used to encrypt messages (put messages into code).

A PRIVATE KEY which you do not make public, and which you use to decrypt messages (decode messages).

The public key gives away lots of information, so public key cryptography requires VERY STRONG cryptographic methods. These methods are based on the existence of very difficult problems called "one way functions." (The existence of these was conjectured by Church and (almost) proved by Cook.) Here's an example which underlies several VERY STRONG public key methods, including the venerable but still unbroken RSA method:

Given two factors of a number, it is easy to multiply them together to get the the product. For example,

$3 \times 7 = 21.$

But given the product number, it is much harder to find the factors. For example,

$21 = 3 \times 7.$

Well, this wasn't too hard, but I'll bet you would be hard pressed to tell me that

$$\begin{aligned}&74068877515858675692517951430592361934474770774867 2\\&81974065794969172976228890022037588025244128056810 3\\&66427833146859564956939017143360568437769525713167 3\\&90005495312574690062280062457161088810028950595 7\end{aligned}$$

can be factored into

$$\begin{aligned}&15094024972934452609983659962770474511394934358673 8\\&38804258766915495884704113536038134442386798911221\end{aligned}$$

$\times$

$$\begin{aligned}&49071654279540277781080595974987892694117655801994 9\\&04744272398370915479278320344512623315863583551217\end{aligned}$$

In the RSA method, the PUBLIC KEY used for putting messages into code is the very large 200 digit integer. The PRIVATE KEY, used for decoding messages, is the pair of 100 digit factors. (By the way, the 100 digit factors are PRIME NUMBERS, so you can't get a factorization by dividing by 3.)

Note: The RSA method was patented by its inventors, Rivest, Shamir and Adelman in 1977. Their patent just expired. But note also: It has recently been revealed that BRITISH INTELLIGENCE actually invented the RSA method in the early 1960's and secretly used it for years.

4. How Secure Is The RSA Method?

Breaking the RSA method involves FACTORING THE PUBLIC KEY into its two prime factors. The empirical and theoretical evidence is that the RSA method is unbreakable if you use a BIG ENOUGH PUBLIC KEY. But the safe key size keeps growing as our knowledge and computer power increase.

The empirical evidence:

The Bank of England used to use an 80 decimal digit public key.

An RSA challenge message with a public key of 129 decimal digits was published in Martin Gardner's "Mathematical Recreations" column in *Scientific American* in August, 1977. Rivest estimated that it would take 40 QUADRILLION YEARS of computer time to factor the public key and break the code. In 1993, Lenstra organized 600 volunteers and 1600 machines from all over the Internet and broke the code in only 8 MONTHS. Today, we could do it with a mere lab full of machines in a few weeks.

The Bank of England now uses a MUCH LARGER PUBLIC KEY.

By the way, the decoded challenge message was, "THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE." This is really deep stuff, isn't it!

The current factorization record is a 155 decimal digit public key in roughly 8.4 years of total computer time (divided among hundreds or thousands of computers, of course). To factor a 310 decimal digit public key would take approximately 1.6 billion times as long, so for the time being, public keys of 300 or more decimal digits are safe.

But even 100 decimal digit public keys are beyond the capacity of Harry Hacker or Script Kiddie to crack.

The theoretical evidence:

All efforts to find really efficient factorization algorithms, or to break the RSA method without actually factoring the public key, have failed.

So have all efforts to find really efficient algorithms for solving NP complete problems (maximally hard problems such as the travelling salesman problem, solvable quickly by inspired guessing, if only one could guess the right answer – for the VERY VERY CURIOUS, the travelling salesman problem and other NP complete problems are described in appendix D).

The theoretical justification for the assertion that public key methods like the RSA method are inherently unbreakable rests on the following widely believed but UNPROVED ASSUMPTIONS:

Breaking the RSA method is equivalent to factoring the public key.

Factorization is not as hard as an NP complete problem like the travelling salesman problem, but is still so hard that it takes EXPONENTIAL TIME to solve.

NP complete problems take exponential time to solve. (This is the famous NP vs P problem, first posed by Cook, which is now one of the Clay Mathematics Institute's 7 Millennium Prize Problems.)

The theoretical foundation for all internet cryptographic security rests on the assumed but UNPROVED truth of these assumptions. It's a bit shaky, isn't it?

5. How Available Are The RSA And Other Strong Cryptographic Methods?

Strong encryption technology is available to almost anyone.

The RSA cryptosystem is sold commercially by RSA Security, Inc.

Anyone can download and use FREE open source versions of the RSA cryptosystem. Phil Zimmerman's PGP (Pretty Good Privacy) program has been available for years. As RSA Security's patent expired in 2001, anyone can now legally use PGP without a license from RSA Security. Needless to say, the FBI and CIA are not happy about this.

For encryption of messages sent over the internet, the industry standard is Netscape's Secure Sockets Layer (SSL). There is now a FREE open source version of the Secure Sockets Layer called Open SSL. The industry standard (and FREE open source) Apache Web Server includes Open SSL.

Other very strong cryptographic systems include:

Elliptic Curve Cryptography, based on the mathematical field of algebraic geometry, and marketed by Hewlett Packard and others.

The NTRU system, based on a mathematical number and ring theoretic method, and marketed by a startup named, appropriately, NTRU. The joke is that NTRU stands for "Number Theorists Are Us."

6. What Are The National Security Implications?

In a word, DIRE.

For years, the U.S. Government tried to prevent the export and distribution of strong cryptographic methods. They failed because:

The cat was already out of the bag. The basic algorithms had been openly published, and strong cryptographic systems were freely sold abroad.

Many did not trust the Government to hold keys in escrow, fearing that the Government would simply spy on all of us.

Do terrorists and criminals use strong cryptographic methods to hide their schemes? YES.

If terrorists tend not use PGP or similar systems to encrypt email, it is only because sending encrypted email attracts unwanted attention.

The terrorists in the Manila plot to blow up 11 US owned commercial airliners used strong encryption to encrypt files on their laptop computers.

The first mob case involving strong cryptography is now being prosecuted. Alleged New Jersey mobster Nicodemo S. Scarfo used PGP to encrypt files which the Government asserts contain records of his loansharking operations. The files were cleverly named "Factors." The FBI used a "keystroke sniffer" to record Scarfo's computer keystrokes and snare his private key. The key was based on his father's federal prison number. Who says FAMILY VALUES are out of fashion?

7. Can We Simply Legislate Against Using Strong Cryptography?

In a word, NO.

Confidential Internet transactions, such as credit card transactions, HAVE TO BE ENCRYPTED because:

> By the very nature of the Internet, it is not only easy, but also legal and NECESSARY for others to listen in to Internet transactions.
>
> Unlike telephone calls, there is no direct dedicated connection for Internet messages. Instead, messages are divided into PACKETS, and each packet is DYNAMICALLY ROUTED from source to destination. One packet may go VIA LONDON, another THROUGH BOSTON, all part of the same message from HERE TO LA.
>
> EVERY COMPUTER at EVERY INTERMEDIATE POINT has to READ EVERY PACKET to see where the packet is addressed, and to forward the packet if necessary.

Possible solutions:

> We could legislate against reading the data contained in the BODY of packets addressed to others, allowing only the reading of the ADDRESS header of the packet.
>
> This would be similar to the way we deal with ordinary surface mail – you can read the address on the envelope, but you can't open the envelope unless it is addressed to you.
>
> But this would be MUCH HARDER TO ENFORCE than current laws outlawing reading other peoples mail or intercepting other peoples telephone messages. (And even defining the meaning of "only read the ADDRESS header" would be very hard.)
>
> I call this the "transparent envelope problem."
>
> Legal note:
>
> > It is ILLEGAL for a PERSON to read the content of packets addressed to someone else. But it is LEGAL for that person's computer to read the packets, and even to temporarily store them for necessary processing (decision making and forwarding). This is what makes enforcement so difficult.
>
> Another approach, tried in Germany, is to outlaw storing ENCRYPTED FILES and sending encrypted email, unless the owner is prepared to provide the decryption key on court order. In the U.S., this might run afoul of the 5th Amendment.

Conclusion:

> As a society, we seem to have made a Faustian bargain. We have traded the economic benefits of Internet commerce, which requires strong cryptography, for added vulnerability to terrorist attack. It is probably too late to turn back the clock.

8. The Real Threat – Digital Steganography.

Steganography is the art and science of hiding messages. The art and science of detecting hidden messages is also part of steganography.

Steganography is real "spy vs spy" stuff.

> It used to involve things like microdots – messages stored on pieces of microfilm the size of a period and inserted into apparently innocuous documents.

Digital steganography works by subtly altering image or music files to hide messages. One scheme:

Start with a 128 color GIF image. The GIF image format allows 256 colors, so you can subtly alter the colors to hide a message. These alterations are not noticeable to the human eye.

The good news: This particular scheme is easy to spot using computerized statistical analysis of the image's color distribution.

The bad news: Schemes involving JPEG images, the most common images on the Internet, are said to be UNDETECTABLE USING CURRENT TECHNOLOGY.

How prevalent is digital steganography?

In a word, VERY.

There are over 100 free downloadable steganography programs on the Internet.

There have been over a million downloads of the technology, from all over the world.

To quote from the New York Times of October 30, 2001,

"Mr. Hosmer's company began looking at millions of digital pictures that were posted on the Internet. They scanned auction sites and pornographic sites, where people can post and download digital images anonymously.

"'We started getting hits,' Mr. Hosmer said, adding that about 0.6 percent of millions of pictures on auction and pornography sites had hidden messages. The messages they found on eBay were encrypted and unreadable, he said."

Do terrorists use digital steganography to hide their schemes? YES

To quote from the same New York Times article again,

"Intelligence officials have not revealed many details about whether, or how often, terrorists are using steganography. But a former French defense ministry official said that it was used by recently apprehended terrorists who were planning to blow up the United States embassy in Paris.

"The terrorists were instructed that all their communications were to be made through pictures posted on the Internet, the defense official said.

"The leader of that terrorist plot, Jamal Beghal, told French intelligence officials that he trained in Afganistan and that before leaving the country for France, he met with an associate of Osama Bin Laden. The plan was for a suicide bomber to drive a minivan full of explosives through the embassy gates."

Can we legislate against steganographic technology?

In a word, NO. The very same techniques used in steganography are used to "watermark" images and music to provide protection against Internet piracy.

If a museum, say, posts images from its collection on the Internet, it watermarks them for protection. If someone steals an image, alters it slightly, and claims, "Hey, I got this image from www.freeimages.com," the hidden watermark provides conclusive proof in court that the image came from the museum's copyrighted collection.

A possible solution:

We could exempt watermarking from a legislative ban on steganography, distinguishing watermarks from other hidden messages by their intended use.

But this would not solve the problem of people posting steganographic images from abroad, unless a broad international consensus against steganography could be formed.

Conclusion:

As a society, we seem to have made another Faustian bargain. We have traded the economic and cultural benefits of posting images and music on the Internet for added vulnerability to terrorist attack. Again, it is probably too late to turn back the clock.

9. Credits.

No Microsoft software was used in the making of this talk.

No monsters were hurt in the making of this talk.

Thankyou very much for listening to me. I will be glad to deal with questions about "How does cryptography affect me as a lawyer" during the question period. And for those who really want to know, I would also have been glad to explain how the RSA method works in greater detail "after class," except that it would take well over an hour. So I have included Appendix B in the printed version for the very curious.

There is a fairly complete list of references in the printed version.

10. References.

Starting Points:

M. Gardner, "A New Kind of Cipher that Would Take Millions of Years to Break", "Mathematical Recreations" column, *Scientific American*, August, 1977. This is the famous column by Martin Gardner that first carried the news of the RSA method to the general public, and brought the FBI to his door.

A. K. Dewdney, "On Making and Breaking Codes: Part I", "Computer Recreations" column, *Scientific American*, October 1988, pp. 144–147. The best short introduction ever to the history of cryptography and the Enigma codes.

A. K. Dewdney, "On Making and Breaking Codes: Part II", "Computer Recreations" column, *Scientific American*, November 1988, pp. 142–145. Includes a short readable description of the DES.

S. Garfinkel, *PGP*: *Pretty Good Privacy*, O'Reilly, 1995.

M. Hellman, "The Mathematics of Public-Key Cryptography", *Scientific American*, August 1979, pp. 146–157. A wonderfully succinct presentation of the ideas of public key cryptography, including the RSA method, by *the Hellman* of Diffie and Hellman.

C. Neville, *Codes and Internet Security*, 1999, http://www.cwnresearch.com/research/

Starting Points for VERY SERIOUS STUDY:

N. Biggs, *Discrete Mathematics*, Oxford, 1987. The BEST reference I know for the basic mathematics involved.

D. Welsh, *Codes and Cryptography*, Oxford, 1990. The BEST reference I know for the serious study of cryptography.

Basic Cryptography References:

Counterpane Internet Security, Inc. Home Page, http://www.counterpane.com (accessed 5/20.02). The Counterpane site includes the following pages:

http://www.counterpane.com/crypto-gram.html (accessed 4/1/02) – an excellent crypto source by Bruce Schneier, founder and Chief Technology Officer of Counterpane.

http://www.counterpane.com/labs.html (accessed 4/2/02) – includes a comprehensive crypto bibliography and many of Schneier's papers in pdf format.

http://www.counterpane.com/publish.html (accessed 5/20/02) – includes a complete indexed list of papers and other Counterpane publications.

W. Diffie and M. Hellman, "New directions in cryptography", *IEEE Trans. Info. Theory* 22 (1976), pp. 644–654. The paper that started it all.

N. Koblitz, *Algebraic Aspects of Cryptography*, Springer Verlag, 1998.

NTRU Cryptosystems, Inc. Home Page, http://www.ntru.com (accessed 2/25/02). An excellent description of the NTRU public key cryptography system, including tutorials and references.

National Institute of Standards and Technology Advanced Encryption Standard (AES) Page, http://csrc.nist.gov/encryption/aes (accessed 5/20/02).

National Institute of Standards and Technology Cryptographic Module Validation (CMV) Program Page, http://csrc.nist.gov/cryptval (accessed 5/20/02).

M. Oskin, F. Chong and I. Chuang, "A Practical Architecture for Reliable Quantum Computers", *IEEE Computer* 35, 1 (January 2002), pp. 79–87. Contains an informative sidebar about quantum computing and cryptography.

L. Preneel, ed., "Factorization of a 512 Bit RSA Modulus", *Lecture Notes in Computer Science*, 1807 (2000), Springer Verlag, 2000.

R. Rivest, A. Shamir and L. Adelman, "Cryptographic communications system and method", U.S. Patent # 4,405,829, filed 1977, granted 1983. The patent on the RSA method.

R. Rivest, A. Shamir and L. Adelman, "A method for obtaining digital signatures and public key cryptography", *Comm ACM* 21 (1978), pp. 120–126. The original paper on the RSA method.

RSA Security, Inc. Home Page, http://www.rsa.com (accessed 2/25/02).

B. Schneier, *Applied Cryptography, protocols, algorithms and source code in C*, 2nd ed., Wiley, 1996.

Cryptographic Analysis and Design References:

J. Daemen and V. Rumen, "The Design of Rijndael, the wide trail strategy", Springer Verlag, 2002. Rijndael (pronounced "Rain Doll") is the system selected to be the Advanced Encryption Standard (AES) by the National Institute of Standards and Technology. J. Daemen is the co-inventor along with V. Rijmen of Rijndael.

B. Schneier, "A Self-Study Course in Block Cipher Cryptanalysis", *Cryptologia*, 24, 1 (January 2000), pp. 18–34. Available in pdf format from http://www.counterpane.com/labs.html (accessed 4/2/02).

B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall and N. Ferguson, *The Twofish Encryption Algorithm, a 128-bit block cipher*, Wiley, 1999.

B. Schneier, D. Wagner and Mudge, "Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)", CQRE '99, Springer Verlag, 1999, pp. 192–203. This paper analyzes the fixes Microsoft made to its Virtual Private Networking software as a result of the serious cryptographic deficiencies revealed in the author's 1998 predecessor paper. The papers in pdf format and the press release are available at http://www.counterpane.com/pptp.html (accessed 5/19/02).

N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner and D. Whiting, "Improved Cryptanalysis of Rijndael", *Seventh Fast Software Encryption Workshop*, Springer-Verlag, 2000.

Cryptography Contacts:

Joan Feigenbaum, CS Department, Yale University, New Haven, CT, http://www.cs.yale.edu/homes/jf/home.html (accessed 5/21/02). She is the Editor-in-Chief of *The Journal of Cryptology, the journal of the International Association for Cryptologic Research (JACR).*

Charles Neville, CWN Research, West Hartford, CT, http://www.cwnresearch.com (accessed 5/21/02). He is the founder and CEO of CWN Research and the author of this document.

Bruce Schneier, http://www.counterpane.com (accessed 5/21/02). He is the founder and Chief Technology Officer of Counterpane.

David Wagner, CS Department, U.C. Berkeley, Berkeley, CA, http://www.cs.berkeley.edu/ daw (accessed 5/21/02).

Peiter Mudge Zatko (Mudge), http://www.atstake.com, (accessed 5/21/02). He is Chief Scientist at @Stake, and the original author of "L0phtCrack", the NT password auditing (and cracking) tool.

Note: None of these people (except for me) have given permission to have their names listed here, so please don't bother them unless you really need cryptographic expertise and are willing to pay for it.

Steganography References:

Johnson & Johnson Technology Consultants, LLC, *Steganography & Digital Watermarking page*, http://www.jjtc.com/Steganography/ (accessed 3/12/02).

N. Johnson and S. Jajodia, "Steganography: Seeing the Unseen", *IEEE Computer* 31, 1 (February 1998), pp. 26–34. Good overview with several examples of messages hidden in images. Available online at http://www.jjtc.com/stegdoc/index2.html (accessed 3/12/02).

N. Johnson, Z. Duric and S. Jajodia, *Information Hiding: Steganography and Watermarking - Attacks and Countermeasures*, Kluwer Academic Publishers, 2000.

S. Katzenbeisser and F. Petitcolas, ed., *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House Books, 2000.

G. Kolata, "Veiled Messages of Terror May Lurk in Cyberspace", *New York Times*, October 30, 2001, p. F1 ff.

F. Petitcolas, *the information hiding home page, digital watermarking and steganography*, http://www.cl.cam.ac.uk/ fapp2/steganography/ (accessed 3/12/02). Contains an excellent pre-2000 annotated bibliography.

Secure Sockets Layer References:

A. Freier, P. Karlton and P. Kocher, *The SSL Protocol, Version 3.0*, Netscape Communications Corp., 1996, http://home.netscape.com/eng/ssl3/draft302.txt (accessed 2/25/02).

The OpenSSL Project Home Page, http://www.openssl.org (accessed 2/25/02).

NP vs P References:

S. Cook. "The complexity of theorem-proving procedures", in *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151 – 158, Shaker Heights, Ohio, 3 – 5 1971.

S. Cook. "The P versus NP Problem", *Clay Mathematics Institute Millennium Prize Problems*, http://www.claymath.org/prize_problems/p_vs_np.pdf, accessed 10/10/2000.

M. Garey and D. Johnson. *Computers and Intractability, a guide to NP-completeness*, W. H. Freeman and Co., San Francisco, 1979.

C. Neville, *The NP vs P Problem*, 2000, http://www.cwnresearch.com/research/

Legal References:

Cornell Law Library Legal Information Institute, US CODE COLLECTION, Title 18, Chapter 119, *Wire and electronic communications interception and interception of oral communications*, http://www4.law.cornell.edu/uscode/18/pIch119.html (accessed 3/7/02).

Cornell Law Library Legal Information Institute, US CODE COLLECTION, Title 18, Chapter 121, *Stored wire and electronic communications and transactional records access*, http://www4.law.cornell.edu/uscode/18/pIch121.html (accessed 3/7/02).

Jones Telecommunications & Multimedia Encyclopedia, *Electronic Communications Privacy Act*, http://www.digitalcentury.com/encyclo/update/ecpa.html (accessed 3/7/2002). A nice summary of the Electronic Communications Privacy Act (ECPA) of 1986.

Public Policy References:

W. Diffie and S. Landau, "September 11th Did Not Change Cryptography Policy", *Notices of the American Mathematical Society*, 49, 4 (April, 2002), pp. 450-454. A wonderfully informed article co-authored by *the Diffie* of Diffie and Hellman, the inventors of public key cryptography, explaining why September 11th has not yet changed cryptographic policy. Available online at http://www.ams.org/notices/ (accessed 3/22/02).

P. Zimmermann, "Cryptography and the Internet", *Scientific American*, October 1998, pp. 110–115.

Appendix A. How Does Cryptography Affect Me as a Lawyer?

A1. What Encryption Should You Use And When Should You Use It.

ALL EMAIL, AND ALL INTERNET TRANSMISSIONS, can be easily and LEGALLY intercepted. So as lawyers, to preserve lawyer-client confidentiality, YOU HAVE A PROFESSIONAL AND LEGAL OBLIGATION TO ENCRYPT ANY CONFIDENTIAL INFORMATION SENT OUTSIDE YOUR OFFICE OVER THE INTERNET. As for method, the industry standard Secure Sockets Layer and IPSEC (Internet Protocol Security) are always good choices.

Secure encryption is tricky to install. For instance, confidential email usually has to be subjected to additional encryption steps for various technical reasons. So you probably need to hire a COMPETENT SECURITY CONSULTANT to set up your encryption systems.

Your clients, if they deal with confidential information belonging to others, have a similar obligation.

Legal note:

> Remember that it is ILLEGAL for a PERSON to read the content of packets addressed to someone else. But it is LEGAL for that person's computer to read the packets, and even to temporarily store them for necessary processing (decision making and forwarding). This is what makes it so difficult for you or your client to prove that someone was illegally eavesdropping.

Some of your clients may need to transfer confidential information over the Internet between offices. For example, a firm with offices in Connecticut and Georgia might need to transfer sensitive database information from one office to another. You should strongly advise your clients in similar situations to encrypt their sensitive internet transmissions.

Passwords for remote login and data transfer SHOULD ALWAYS BE ENCRYPTED. Usually, whatever software you are using for remote access includes the necessary encryption modules.

In terms of legal liability, I know I am a non-expert talking to the experts here, but I would say that you or your clients would be adequately protected by using ANY commercial encryption technology, or by using ANY industry standard free open source encryption technology.

I am embarrassed to caution a group of lawyers about reading and abiding by the licensing agreements for all software, but please do it. In particular, most free open source software can be freely used for any legal purpose, but some is restricted to PERSONAL NON-COMMERCIAL USE. For example, free PGP now comes with this restriction.

A2. What Should You Not Encrypt.

The general rule is,

> Encrypt external transmissions. Don't encrypt files stored locally on disk.

Some security experts recommend encrypting ALL sensitive information, even if it is stored locally on disk, as a precaution against unauthorized intruders (hackers). But you have to be careful about this:

> If you lose the keys, you lose the information. Storing the keys on your computer defeats the purpose of encrypting the data in the first place.
>
> And as officers of the court, or in the case of your clients simply as honest citizens, you have to be prepared to produce the keys on subpoena.

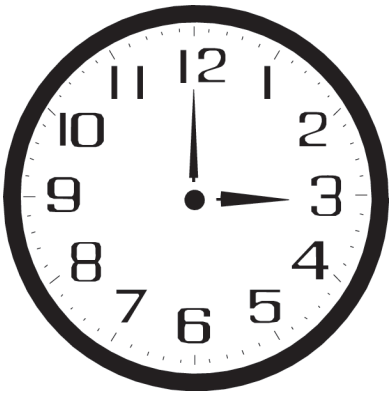
The difference is that software for encrypting external transmissions, such as the Secure Sockets Layer, only encrypts the data while it is being sent, and automatically decrypts it at the receiving end. So you never have to worry about keeping keys secure, and you never have to worry about producing the keys on demand.

Appendix B. How the RSA Method Works.

B1. Modular Arithmetic.

The RSA method involves modular arithmetic. Modular arithmetic is sometimes called "clock arithmetic," and you probably saw it in grade school.

For example, consider the clock below:



12 hours from 3 o'clock, it will be 3 o'clock again, and 13 hours from 3 o'clock, it will be 4 o'clock. Mathematically:

3 o'clock = 3 o'clock + 12 hours - 12 hours.
4 o'clock = 3 o'clock + 13 hours - 12 hours.

In clock arithmetic, we are allowed to add or subtract any multiple of 12 to bring the time back into the range from 1 o'clock to 12 o'clock.

In modular arithmetic, we would say:

$3 = 3 + 12 \mod 12.$
$4 = 3 + 13 \mod 12.$

In mod 12 arithmetic, just as in clock arithmetic, we are allowed to add or subtract any multiple of 12. The only difference is that we usually bring the answer back into the range from 0 to 11, rather than 1 to 12. Thus in mod 12 arithmetic,

$0 = 3 + 9 \mod 12,$

whereas in clock arithmetic,

12 o'clock = 3 o'clock + 9 o'clock.

We can do modular arithmetic with respect to a modulus other than 12. Here are some simple examples:

$2 = 3 + 9 \mod 10,$
$1 = 4 + 2 \mod 5,$

because

$2 = 12 - 10 = 3 + 9 - 10,$
$1 = 6 - 5 = 4 + 2 - 5.$

We can also multiply in modular arithmetic. For example,

$$1 = 7 \times 3 \bmod 10$$

because

$$1 = 21 - 20 = 7 \times 3 - 20,$$

and 20 is an integral multiple of 10.

This example is particularly interesting because the answer is 1. We say that 3 is the inverse of 7 mod 10, or

$$3 = 7^{-1} \bmod 10$$

because

$$1 = 7 \times 3 \bmod 10,$$

just as $1/7$ is the inverse of 7 in ordinary arithmetic because

$$1 = 7 \times 1/7.$$

It turns out that calculating inverses in modular arithmetic is very important in the RSA method.

B2. Some Classical Mathematics.

The mathematics behind the RSA method was invented

By EUCLID circa 300 BC.
By EULER circa 1750 AD.

Euclid proved:

There are infinitely many prime numbers. Remember that a whole number p is prime if it can only be evenly divided by itself and 1. Thus 3 and 5 are prime, but 4 and 15 are not.

Euclid also invented (or at least codified):

The "Euclidean algorithm", which is essentially the familiar algorithm for long division we learn in grade school. The Euclidean algorithm allows us to easily find the greatest common divisor of a pair of numbers, without factoring them into a product of primes. For example:

The greatest common divisor of 12 and 30 is 6.
The greatest common divisor of 5207 and 5461 is 127.

I won't show you how the Euclidean algorithm can be applied to calculate this; just take my word for it.

We use the Euclidean algorithm to calculate inverses in modular arithmetic. I won't show you how to do this either. Consult the book on discrete mathematics by Biggs cited in the references for a complete explanation and examples.

Euler discovered the laws of exponents for modular arithmetic:

In ordinary arithmetic and algebra,

$$a^{r+s} = a^r \times a^s.$$

For example,

$$4^5 = 4^2 \times 4^3,$$

This fundamental law of exponents is still valid in modular arithmetic, but Euler showed we can say more:

$$a^{r+s} \bmod N \neq a^r \times a^s \bmod N,$$

but

$$a^{r+s} \bmod \phi(N) = a^r \times a^s \bmod N.$$

Here, $\phi(N)$ is a function called the "Euler ϕ function." If N is the product of two large primes,

$$N = p \times q,$$

then, Euler showed,

$$\phi(N) = (p-1) \times (q-1).$$

In the RSA method, we raise numbers to a large power in mod N arithmetic to encode the message, and we raise the coded message numbers to another large power in mod N arithmetic to decode the message. The powers are chosen so their product in mod $\phi(N)$ arithmetic is 1. The reason one needs to know the factorization of N to decode the message is that the only computationally feasible way known to calculate $\phi(N)$, and thus get the decoding power from the encoding power, is to know the factorization of N.

B3. A Concrete Example, the First RSA Message.

The first published example of the RSA method was given by Rivest, Shamir and Adelman in the *Communications of the ACM* in 1978. (It actually first appeared in Martin Gardner's *Scientific American* column in 1977, but he was quoting from the preprint.) Here it is.

The message is "ITS ALL GREEK TO ME". The message is converted to a sequence of four digit numbers this way:

Space = 00, A = 01, B = 02, etc.

The sequence of four digit numbers is

M = 0920 1900 0112 1200 0718 0505 1100 2015 0013 0500 etc.

Here, the symbol "M" stands for "Message." For instance, the "I" of "ITS" is represented by 09, the "T" by 20, the "S" by 19, and the space following "ITS" by 00. As a check, note that the "A" of "ALL" is represented by 01, as it should be.

The pair of primes are $p = 47$ and $q = 59$. (This is a "toy example." For a real message, such as your credit card number, the primes would be MUCH LARGER.) The "encryption modulus" $N = pq = 2773$. The Euler function $\phi(N) = (p-1)(q-1) = 2668$.

The public key is the modulus N and the "encoding exponent" $e = 17$. The private key is the modulus N and the decoding exponent $d = 157$. (You can see that I oversimplified a bit in the main body of the talk when I said the public key was the encryption modulus N and the private key was the pair of primes, p and q.)

Rivest, Shamir and Adelman could calculate d from e and N because they knew the values of the prime factors p and q, and thus could compute the Euler ϕ function $\phi(N) = (p-1)(q-1)$. The value of d was calculated to be 157 so that

$$1 = e \times d \bmod \phi(N).$$

To put the message M into code, Rivest, Shamir and Adelman used the formula

$$C = M^e \bmod N.$$

Here, the symbol "C" stands for "Code." They applied this formula to each four digit number making up M. Thus

$$\begin{aligned} C &= 0920^{17} \bmod 2773,\ 1900^{17} \bmod 2773, \text{ etc.} \\ &= 0948\ 2342 \text{ etc.} \end{aligned}$$

To decode the coded message, they used the formula

$$M = C^d \bmod N,$$

applied to each four digit number making up C. Thus

$$\begin{aligned} M &= 0948^{157} \bmod 2773,\ 2342^{157} \bmod 2773, \text{ etc.} \\ &= 0920\ 1900 \text{ etc.} \end{aligned}$$

(Of course they used a computer to do this. They also were helped by the fact that each time they multiplied a four digit number like 0948 by itself, they could reduce mod 2773 to get another four digit number. In this way, they avoided extremely large numbers.)

B4. Why This Works.

This works because $1 = ed \bmod \phi(N)$, and because of Euler's formula,

$$a^{ed} \bmod {}^{\phi(N)} = a^{ed} \bmod N.$$

Thus,

$$\begin{aligned} C^d \bmod N &= (M^e)^d \bmod N = M^{ed} \bmod N = M^{ed} \bmod {}^{phi(N)} \bmod N \\ &= M^1 \bmod N = M. \end{aligned}$$

In other words, raising the coded message C to the decoding exponent d in mod N arithmetic results in the original message M.

Appendix C. Equivalent Forms of Computation and the Invention of the Stored Program Computer.

B1. Princeton 1937 – 1939.

By 1937, several very different formal systems for computation had been invented. Prominent among these were,

Goedel's "Logic + arithmetic". The famous Goedel undecideability and incompleteness theorems were proved using this formal system. (Goedel's theorems stunned the world by showing that there were some mathematical statements which could neither be proved nor disproved.)

Turing's "Turing machines". Turing machines were the first actual designs for a computer, and they are still used for theoretical analysis due to their simplicity.

Post's "λ calculus". The artificial intelligence language LISP is based on the λ calculus.

Between 1937 and 1939, Goedel, Turing, Post, Church, Von Neumann and others gathered at the Institute for Advanced Study to try to make sense out of the babel of formal computing systems. They found that what could be decided or computed in one system could be decided or computed in each of the other systems. In other words, the systems were all equally powerful and equally powerless because they could each compute exactly the same things.

They finally came to understand why each of these apparently very different systems was exactly as powerful as the others. It turned out that each system could simulate each of the others. Perhaps this is most easily understandable if we start with Turing machines, which are very recognizably computers, and consider the λ calculus, which looks very much like a computer language. In modern terms, Turing machines could decide and compute exactly what the λ calculus could compute, because one could write an "interpreter" to run λ calculus programs on Turing machines. Similarly, one could write a program in the λ calculus to simulate a Turing machine. (Today, we sometimes give beginning computer science students a project to write a program in Pascal or Java to simulate a Turing machine.)

B2. Consequences of Princeton 1937 – 1939.

Out of this effort came some very important ideas:

The Church – Turing thesis that all forms of computation are equally powerful (or powerless).

The idea of compilers or interpreters to run a program written in a computer language, such as the λ calculus, on a computing machine, such as a Turing machine.

The idea of simulating or emulating one system with very different architecture in a different system. Today, we refer to this as writing a virtual machine to emulate one machine in another.

The idea that there was really no difference between programs and data. For example, what was a program (in the λ calculus, say) was input data for a Turing machine running its emulation program.

And out of the last insight, that there was really no difference between programs and data, came, in John Von Neumann's brilliant mind, the invention of the stored program computer.

In fact, each of these insights motivated important advances in computer science:

The Church – Turing thesis formed the intellectual basis for the first artificial intelligence research. The idea was, the human brain is simply another sort of computer, a very different

computer, but a computer non-the-less. By the Church – Turing thesis, what the mind can do (compute) can be done (computed) by an ordinary stored-program computer. This statement actually appears explicitly in the original grant, submitted to the Office of Naval Research after World War II, for the first artificial intelligence research.

Compilers, interpreters, and higher level computer languages such as FORTRAN and C came out of the idea of running λ calculus programs on Turing machines.

Writing virtual machines to emulate one computer on another computer with a very different architecture came out of simulating one formal system within another. This idea is COMMERCIALLY VERY IMPORTANT today. For example, Advanced Micro Devices uses a virtual machine, or the micro-coded equivalent, to emulate an Intel Pentium chip on the very different AMD Athalon chip.

Acknowledgments.

Thanks to

Steve Buck
Kent Hughes
Dave Jones
Cliff Pelletier

for reading and commenting on this paper.