CINDY A. COHN, ESQ.; SBN 145997
McGLASHAN & SARRAIL
Professional Corporation
177 Bovet Road, Sixth Floor
San Mateo, CA 94402
Tel: (415) 341-2585
Fax: (415) 341-1395

LEE TIEN, ESQ.; SBN 148216
1452 Curtis Street
Berkeley, CA 94702
Tel: (510) 525-0817

Attorneys for Plaintiff
Daniel J. Bernstein

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

DANIEL J. BERNSTEIN)
) C 95-00582 MHP
Plaintiff,)
) DECLARATION OF
v.) BRUCE SCHNEIER
))
UNITED STATES DEPARTMENT OF)
STATE et al.,))
Defendants.)
)
)

I, BRUCE SCHNEIER declare as follows:
1. I am president of Counterpane Systems, a consulting firm specializing in cryptography and computer security. I am the author of Applied Cryptography: Protocols, Algorithms, and Source Code in C, the second edition of which was published in 1996 by John Wiley & Sons, and of E-Mail Security, published in 1995 by John Wiley & Sons. I am a contributing editor to the computer programming magazine Dr. Dobb's Journal where I have edited the "Algorithm Alley" column. I am a member of the board of directors of the International Association for Cryptologic Research. I have personal knowledge of the facts set forth herein, unless otherwise indicated, and if called as a witness could and would so testify. I executed this Declaration at Stanford, California.

BASICS OF CRYPTOGRAPHY

2. The aim of encryption is to turn an otherwise intelligible message into gibberish, so that a person who intercepts the message cannot read it. Moreover, the intended recipient should be able to take that gibberish and turn it back into an intelligible message. The science of doing this sort of thing is called cryptography.
3. An example: Alice is the sender, Bob is the receiver, and Eve is the eavesdropper. Alice wants to send a message to Bob. She takes the message, also called the plaintext, and encrypts it. The encrypted message is called the ciphertext. Alice then takes the ciphertext and sends it to Bob. Eve, who intercepts the ciphertext in transmission, can't read it. But Bob, who legitimately receives the ciphertext, is able to decrypt it and recover the plaintext message. This model presupposed some arrangement between Alice and Bob that Eve is not privy to.

4. Encryption is based on two things: an algorithm and a key. An algorithm in cryptography is a mathematical transformation. It transforms plaintext into ciphertext, and it transforms ciphertext back into plaintext. When Alice encrypts a message, she uses an encryption algorithm to transform the plaintext into ciphertext. When Bob decrypts a message, he uses the corresponding decryption algorithm to transform the ciphertext back into plaintext. This encryption transformation can be preformed on the letters of the message or on the digital representation of the message. Prior to the computer age, encryption was primarily done on alphabetic characters. The encryption algorithm would turn the characters of the plaintext message into other characters, or numbers, or symbols. Then, the decryption algorithm would convert them back again. Recently, messages to be encrypted either originate from computers or are typed into computers. In this case, the computer representation of the message (the information bits themselves) are encrypted and decrypted. This generalization means that any digital data can be encrypted: text messages, computer programs, digital audio, digital video, etc.
5. Until very recently (1883 or so), the security of this operation was based on keeping the algorithm secret. If Eve knew the algorithm, she could decrypt the ciphertext just as easily as Bob. But if Alice and Bob keep the details of the algorithm to themselves, Eve won't know what to do. The best she can do is to try to figure out what the algorithm is. This is called cryptanalysis, and there was a whole lot of it going on during World Wars I and II.
6. There are problems with requiring the algorithm to remain secret. One, it is not easy to do. The militaries of the world spend a lot of time and energy keeping their algorithms secret, and still details leak across borders. If the algorithm is embedded in a piece of military hardware--e.g. a tactical radio, a computer modem, a satellite receiver--eventually the enemy will capture or steal one and reverse-engineer the algorithm. If the algorithm is in a commercial piece of software, it only takes a few hours for a clever programmer to disassemble the program and reverse-engineer the algorithm. And once the details of the algorithm ever become public, the security of the entire system is lost forever.
7. Two, if this type of system is going to work for everyone, then each pair of people will need a unique algorithm. If there are 100 people in a network, that means that there must be almost 5000 different algorithms. Each person has to know 100 of them. And he has to keep all of them secret from everyone else. Designing an encryption algorithm that is secure from cryptanalysis is a difficult task; designing 5000 of them is absurd.
8. We can get around these problems with the second basis of cryptography: the key. A key is a random string of ones and zeros (called a bit string)--sometimes a number in binary, sometimes an ASCII value, or sometimes a word or phrase that gets converted to a bit string--that is used in conjunction with an algorithm. An algorithm can take any one of a large number of possible keys. Each different key causes the algorithm to work in a slightly different way. Only two people with the identical key can encrypt and decrypt messages. Someone with one key cannot decrypt messages encrypted with a different key.
9. Think of a lock. The world doesn't need a new lock design for every front door. It's enough to have one design, and hundreds of thousands of different keys. Even if two people are using the same lock, they can't open each other's doors unless they have identical keys.
10. The important aspect of this process is that the security of the system rests primarily within the key. If Eve does not have the exact key that Alice and Bob are using, she cannot decrypt the message. We assume that Eve has complete knowledge of the algorithm. We assume that Eve has a copy of the software that Alice and Bob are using to encrypt messages. (If it is mass-market software, this is a perfectly reasonable assumption.) We can even assume that Eve knows what Alice and Bob are encrypting. (This is not always the case, but a prudent cryptographer will assume that Eve has access to some of the plaintext. In this case, she is attempting to learn the key so that she can decrypt other messages. This was true during World War II, where the British often had the plain text of weather reports and used them to determine the daily German Enigma key so that they could read more interesting messages.) Without the key, Eve cannot decrypt the message.
11. Now, the whole process of encryption is a whole lot easier. Alice and Bob do not have to keep an algorithm secret; they only have to keep a particular key secret. This key might only be twenty or thirty digits, and a whole lot easier to keep secret than an entire software program. They can choose one algorithm, one that has been published and is known to be secure, and not have to design their own. If their message key is compromised, it only compromises the messages encrypted with that key, not all messages encrypted with that algorithm. They can change the key and keep on encrypting messages; they don't have to recall all the

software and design a new algorithm. And an entire network of users can communicate securely with one another using just one algorithm and many different keys.
12. This is where encryption is today. We encrypt messages using algorithms that are public and known--the Data Encryption Standard or DES, IDEA, etc.--and secret keys. Different users use different keys; smart users change keys weekly, daily, or even more frequently. Assuming those algorithms are secure, then messages encrypted with them are as secure as the key.
13. The difficulty lies in the assumption that an algorithm is secure. In general, it is impossible to prove the security of an encryption algorithm. (There is one exception known as a "one-time pad" that is not germane to this discussion.) What we have is a handful of algorithms that are believed to be secure. "Believed" is the key word in that sentence. In cryptography, it is only possible to prove that an algorithm is insecure. You can only say is that no one can break the algorithm, yet. You can say that these particular people, all intelligent cryptographers, have spent thousands of man-hours trying to break the algorithm, and they have not succeeded. You can say that there is no known attack that can break the algorithm. But you cannot say that someone won't figure out how to break the algorithm tomorrow. You can't even say that one of those military agencies mentioned above doesn't already know how to break the algorithm, because if they did they would not go public about it.
14. For the most part, the encryption algorithms used in commercial products are all published algorithms, and they are all algorithms that have undergone years of public scrutiny. While this doesn't prove anything, it does instill some confidence in the algorithms. In some cases, it puts bounds on the level of security the algorithm can deliver. This means we can make statement about the algorithm's resistance to specific known attacks. While it is certainly possible that new, unknown, attacks could be developed, it is more likely that an attacker would use known attacks.

CRYPTOGRAPHY AS A COMPUTER APPLICATION

15. Cryptography is well-suited for computers. Today almost all cryptography is done by computers or dedicated computer chips. The reason for this is simple: modern cryptographic algorithms are a combination of mathematical and logical operations on bits or groups of bits. This is exactly the kind of thing that computers are so well suited for. In a computer, all information is represented as bits, regardless of what it is or how it is used.
16. An example of a basic mathematical and logical operation which is used in cryptography, but is not itself a cryptographic algorithm, is Euclid's Algorithm. This algorithm was first written down by Euclid in 300 B.C., and describes a method for finding something called a "greatest common divisor" between two numbers. This is the largest factor that they share in common. For example, the greatest common divisor of 18 and 45 is 9; the greatest common divisor of 15 and 28 is 1. Here is Euclid's algorithm being used to find the greatest common divisor of 6 and 22:

22 = 3 * 6 + 4
6 = 1 * 4 + 2
4 = 2 * 2 + 0

The final 0 means the algorithm terminates, and the solution is the third number in the last row: 2. This is Euclid's algorithm written in the C programming language.

```
int gcd (int x, int y) { int g; if (x < 0) x = -x; if (y < 0) y = -y;
if (x + y == 0) ERROR; g = y;
while (x > 0) {g = x; x = y % x; y = g;} return g; }
```

Running the above program with the inputs 6 and 22 yields the same answer: 2:
17. A more complex algorithm is the IDEA encryption algorithm. It takes a 64 bit block of plaintext and turns it into a 64-bit block of ciphertext, with the help of a 128-bit key. Remember, computers represent everything using bits, so anything in a computer can be encrypted in this manner. IDEA encrypts these bits in groups of 64 bits; this is called a block. By using basic mathematical operations like the exclusive-OR logical operation, addition, and modular multiplication, the computer combines the key and the plaintext message to produce ciphertext. It is somewhat like shuffling cards over and over again, but the cards change because you're shuffling them. A full explanation of how IDEA works is provided in my book, APPLIED

CRYPTOGRAPHY (Second Edition), on pages 319-325. It's too involved to discuss here. It is possible to implement IDEA by hand, but it is enormously easier to do it on computer. Remember, computers already represent data in terms of bits, and they naturally do the mathematical operations required by the IDEA algorithm. In my book APPLIED CRYPTOGRAPHY, available in foreign countries such as the U.K. Singapore, Japan, Iran, South Africa, Germany, Russia, Korea, and Australia, I published source code for IDEA written in the C programming language. The State Department has declared that my book is in the public domain and thus freely exportable without a license. Both the description and the C code describe the same thing. They convey the same information. One is easier to read, and the other is easier to embed in a computer program. Any competent programmer can translate one into the other; no cryptographic skills are required. C code for the IDEA algorithm is also freely available on the Internet at archive sites in the U.K., Finland, Germany, New Zealand, and Italy. Internet users from all over the world can download C source code from these sites.

THE SCIENCE OF CRYPTOGRAPHY

18. Modern cryptography is a mathematical science. Its practitioners often have doctoral degrees in either mathematics or theoretical computer science. Encryption algorithms are mathematical transformations of plaintext to ciphertext. Techniques to analyze and possibly break algorithms are mathematical in nature. Mathematical areas such as number theory, complexity theory, and information theory are at the heart of modern cryptography.

19. A good example is the RSA public-key cryptosystem. This cryptosystem uses a different kind of "lock" than we're accustomed to. Normally, you lock and unlock your front door with the same key. If you lose that key, the person who finds it can both lock and unlock your door. With RSA, however, there is a pair of keys, one "public" and one "private: you encrypt with the public key, and decrypt with the private key.

20. The big advantage of this "asymmetric" system is that Alice and Bob can each publish their public key. If Alice uses Bob's public key to encrypt her message to him, then Bob decrypts Alice's message using his private key. Alice can't decrypt a message encrypted with Bob's public key if she doesn't know his private key.

21. The trick to RSA is that although in theory one could figure out the private key from the public key, the mathematical relationships between them are believed to be impossible for computers to solve without taking a very long time. For example, the RSA encryption algorithm looks like this:

P^E mod N = C

This is a mathematical relationship between the key (E and N), the plaintext (P), and the ciphertext (C). Additional mathematical relationships determine how E and N are constructed, as well as the construction of another key: D.

22. Cryptographers believe that the security of the RSA algorithm is based on the difficulty of factoring the number N. Factoring a number means finding the prime numbers that can be multiplied to produce that number; factoring 60, for instance, means showing that 60 = 2 * 2 * 3 * 5. This looks easy, but suppose you multiply two very large prime numbers, each of more than 100 digits? Multiplying these numbers can be done quickly on a computer, but taking that product and factoring it to recover the two 100-digit prime numbers takes much longer. In 1994, a 129-digit number was factored using 600 people and 1600 computers over a period of eight months.

23. The factoring problem is one of the oldest in mathematics. The factoring of the 129-digit number described above used mathematical theory that was not state-of-the-art even at that time. The computation would have taken about one-tenth of the time using the best theory we know of today. Thus, a mathematical problem is at the heart of this encryption algorithm.

24. It is just as natural to look at the above algorithm as computer code. In the LISP programming language, RSA encryption might look like:

```
(DEFFUN RSA-ENCRYPT (P E N)
(MODULO (POWER P E) N))
```

25. In the C programming language, RSA encryption might look like:

```
number RSA_ENCRYPT (number *P, *E, *N) {return modulo (power (P,E), N) }
```

26. All of these three representations can be read by programmers and mathematicians, and all convey the same information. Cryptographers use different representations in different contexts. In my book, APPLIED CRYPTOGRAPHY, I used both. Some algorithms I explained using mathematical notation; others I explained using C programming notation. Both notations were intended to be read by humans, and I made my choice based on which notation could best explain the algorithm to my readers.

CRYPTOGRAPHY AND EXPORT CONTROLS

27. The ITAR is deliberately vague on the subject of cryptographic research. In APPLIED CRYPTOGRAPHY, I devoted about 100 pages to source code for various cryptographic algorithms. I tried to convince my publisher, John Wiley & Sons, to put the same source code on a computer disk to be distributed with the book. Many books in the area of computers, computer science, and programming are offered with floppy disks containing programs referenced in those books. My publisher initially decided to pursue this option--books with floppy disks or CD-ROMs included tend to sell better than books without--but then backed off when informed about the ITAR. My publisher was unsure about the law, and decided that it was not worth the risk. Instead, I offer a set of disks containing many cryptographic algorithms in source code which I will mail only to addresses within the United States and Canada. Thus, I have been affected by export controls on cryptography.

CRYPTOGRAPHERS AS SCIENTISTS

28. Cryptography is a science of making and breaking algorithms. Algorithms are proposed one year and broken the next. For all practical purposes, it is impossible to prove the security of an algorithm. All that you can say is: "I do not know how to break this algorithm, and all of these smart people do not either.
29. It is this reliance on peer review that makes publication and dissemination of algorithms vital to the science of cryptography. All sciences require the constant sharing, evaluating, and criticizing of new ideas, but cryptography has by its very nature an adversarial component. Cryptographic algorithms must be evaluated by other cryptographers, over the course of years, and in public settings.
30. An example is the SAFER algorithm. It was created by James Massey in Switzerland, and published in 1994. That publication included both a description of the algorithm and source code in PASCAL. C source code for SAFER and posted to the Internet. A group of cryptographers invented an alternative key schedule for the algorithm which was also posted to the Internet. Many cryptographers around the world studied the algorithm; these results were also published. More recently, Lars Knudsen of Denmark has proposed a modification to the algorithm designed to correct weaknesses that have been found. His work was completed while he was teaching at a Belgian university and was published in 1995. Source code with Knudsen's modification has already been published on the Internet.
31. Products that use proprietary algorithms, those that are not released to the academic community or to the public, are generally assumed to be weak. Popular software products that try to do this anyway have their cryptography anonymously reverse-engineered and posted to the Internet. (Two examples of this are the algorithms RC2 and RC4.) The reputation of cryptographers is based on what they publish and distribute, not on what they do in secret.
32. Publication means publication of the exact specifications of an algorithm. Sometimes the algorithm is specified in mathematical description. Sometimes it is specified in programming code. Sometimes it is written as a diagram. Sometimes multiple ways are used.
33. The U.S. government specified the DES algorithm mathematically and graphically the moment it was first proposed as a Federal Information Processing Standard. I published source code for DES in Applied Cryptography. The IDEA algorithm, written in Switzerland, was specified in words, pictures, and PASCAL code. I published my own algorithm Blowfish in Dr. Dobb's Journal in words, pictures, and PASCAL code. RC2 and RC4 were first publicly distributed in C code; descriptions followed later.
34. Source code is an especially important dissemination tool because it is exact; it is not subject to interpretation. Cryptographers often publish "reference implementations" of their algorithms. These are meant to be benchmarks against which other implementations are verified. If a cryptographer wants to study an algorithm, he often tests his own code against the reference implementation to ensure that he implemented the algorithm correctly. Reference implementations for many algorithms are available on the Internet, worldwide, for free.
/ / / /