Security Engineering

Lesson 9

Cryptographic Algorithms: Mathematical Background, DES

Spring 2010

Dr. Marenglen Biba

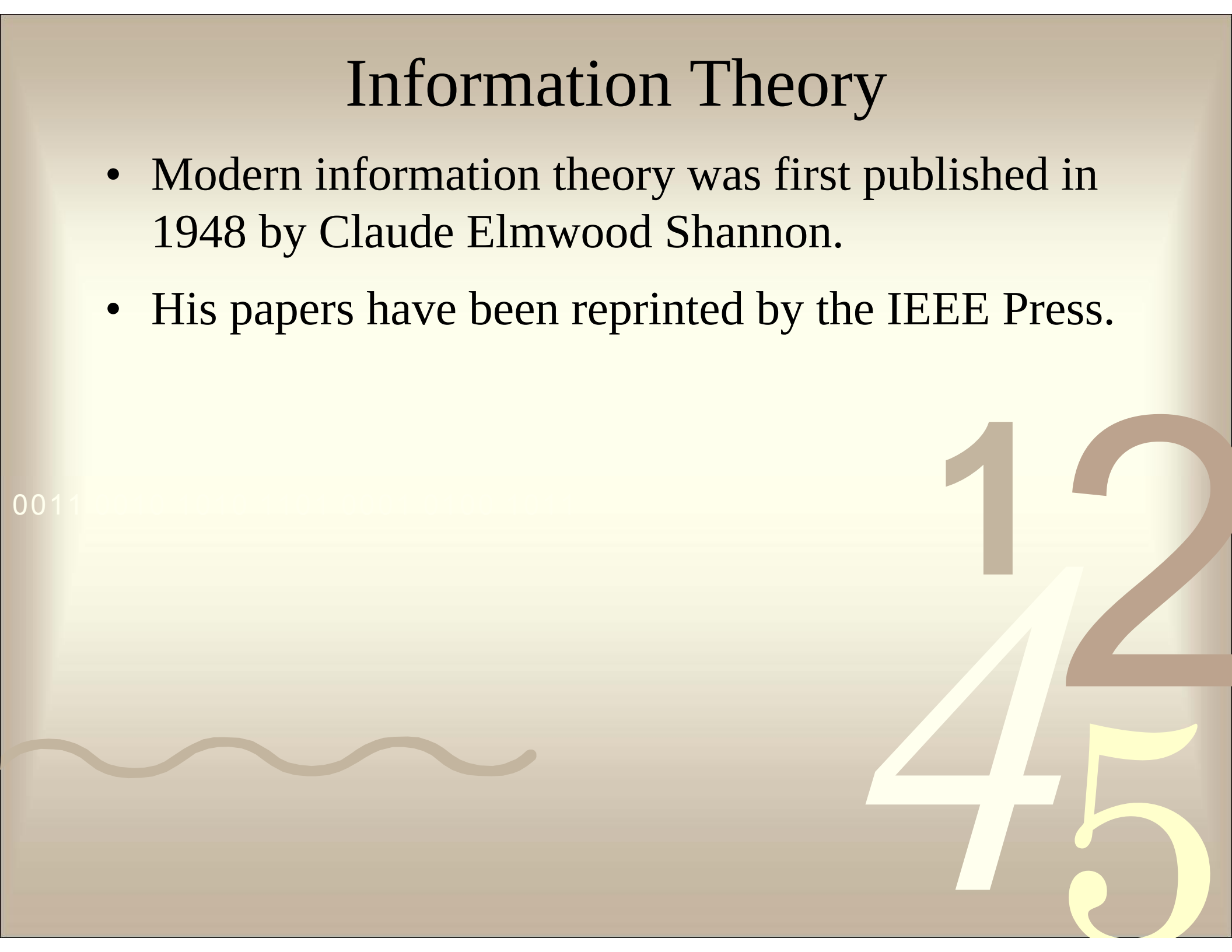
Outline

Mathematical Background

- Information Theory
- Complexity Theory
- Number Theory
- Factoring
- Prime Number Generation

Information Theory

- Modern information theory was first published in 1948 by Claude Elmwood Shannon.
- His papers have been reprinted by the IEEE Press.



Claude Elwood Shannon

- **Claude Elwood Shannon** (April 30, 1916 – February 24, 2001), an American mathematician, electronic engineer and geneticist, is known as "the father of information theory".
- Shannon is famous for having founded information theory with one landmark paper published in 1948. But he is also credited with founding both digital computer and digital circuit design theory in 1937, when, as a 21-year-old master's student at MIT, he wrote a thesis demonstrating that electrical application of Boolean algebra could construct and resolve any logical, numerical relationship. It has been claimed that this was the most important master's thesis of all time.

Entropy and Uncertainty

- Information theory defines *the amount of information in a message as the minimum number of bits needed to encode all possible meanings of that message, assuming all messages are equally likely*.
- For example, the day-of-the-week field in a database contains no more than 3 bits of information, because the information can be encoded with 3 bits:

  000 = Sunday

  001 = Monday

  010 = Tuesday

  011 = Wednesday

  100 = Thursday

  101 = Friday

  110 = Saturday

  111 is unused

- If this information were represented by corresponding ASCII character strings, it would take up more memory space but would not contain any more information.

Entropy and Uncertainty

- Formally, the amount of information in a message *M* is measured by the entropy of a message, denoted by H(*M*).
- The entropy of a message indicating “sex” in a database is 1 bit; the entropy of a message indicating the day of the week is slightly less than 3 bits.
- In general, the entropy of a message measured in bits is $\log_2 n$, in which *n* is the number of possible meanings. This assumes that each meaning is equally likely.

Entropy and Uncertainty

- The entropy of a message also measures its uncertainty.
- This is the number of plaintext bits needed to be recovered when the message is scrambled in ciphertext in order to learn the plaintext.
- For example, if the ciphertext block “QHP*5M” is either “MALE” or “FEMALE, ” then the uncertainty of the message is 1.
  - A cryptanalyst has to learn only one well-chosen bit to recover the message.

Rate of a Language

- For a given language, the rate of the language is

  $r = \mathrm{H}(M)/N$

  in which N is the length of the message.
- The rate of normal English takes various values between 1.0 bits/letter and 1.5 bits/letter, for large values of N.
- Shannon said that the entropy depends on the length of the text.
  - Specifically he indicated a rate of 2.3 bits/letter for 8-letter chunks, but the rate drops to between 1.3 and 1.5 for 16-letter chunks.

Absolute rate of a language

- Absolute rate of a language is the maximum number of bits that can be coded in each character, assuming each character sequence is equally likely.
- If there are *L* characters in a language, the absolute rate is:

  $R = \log_2 L$
- This is the maximum entropy of the individual characters.
- For English, with 26 letters, the absolute rate is $\log_2 26$, or about 4.7 bits/letter.
- It should come as no surprise to anyone that the actual rate of English is much less than the absolute rate.
  - Natural language is highly redundant.

Redundancy of a language

- The redundancy of a language, denoted *D*, is defined by:

  $D = R - r$

- Given that the rate of English is 1.3, the redundancy is 3.4 bits/letter. This means that each English character carries 3.4 bits of redundant information ☺.
- An ASCII message that is nothing more than printed English has 1.3 bits of information per byte of message.
- This means it has 6.7 bits of redundant information, giving it an overall redundancy of 0.84 bits of information per bit of ASCII text, and an entropy of 0.16 bits of information per bit of ASCII text.
- The same message in BAUDOT code, at 5 bits per character, has a redundancy of 0.74 bits per bit and an entropy of 0.26 bits per bit. Spacing, punctuation, numbers, and formatting modify these results.

Security of a Cryptosystem

- Shannon defined a precise mathematical model of what it means for a cryptosystem to be secure.
- The goal of a cryptanalyst is to determine the key *K*, the plaintext *P*, or both.
  - However, he may be satisfied with some probabilistic information about *P*: whether it is digitized audio, German text, spreadsheet data, or something else.

Security of a Cryptosystem

- In most real-world cryptanalysis, the cryptanalyst has some probabilistic information about *P* before he even starts.
- He probably knows the language of the plaintext. This language has a certain redundancy associated with it.
- If it is a message to Bob, it probably begins with “Dear Bob.” Certainly “Dear Bob” is more probable than “e8T&g [, m.”
- The purpose of cryptanalysis is to modify the probabilities associated with each possible plaintext.
- Eventually one plaintext will emerge from the pile of possible plaintexts as certain (or at least, very probable).

Perfect secrecy

- There is such a thing as a cryptosystem that achieves **perfect secrecy**: *a cryptosystem in which the ciphertext yields no possible information about the plaintext (except possibly its length).*
  - Shannon theorized that it is only possible if the number of possible keys is at least as large as the number of possible messages.
  - In other words, the key must be at least as long as the message itself, and no key can be reused.
- In still other words, the one-time pad is the only cryptosystem that achieves perfect secrecy.
- Perfect secrecy aside, the ciphertext unavoidably yields some information about the corresponding plaintext.
- *A good cryptographic algorithm keeps this information to a minimum; a good cryptanalyst exploits this information to determine the plaintext.*

Cryptanalysis and Redundancy

- Cryptanalysts use the natural redundancy of language to reduce the number of possible plaintexts.
- The more redundant the language, the easier it is to cryptanalyze.
- This is the reason that many real-world cryptographic implementations use a compression program to reduce the size of the text before encrypting it.
- Compression reduces the redundancy of a message as well as the work required to encrypt and decrypt.

Entropy of a cryptosystem

- The entropy of a cryptosystem is a measure of the size of the keyspace, K.
- It is approximated by the base two logarithm of the number of keys:

$$H(K) = \log_2 K$$

- A cryptosystem with a 64-bit key has an entropy of 64 bits; a cryptosystem with a 56-bit key has an entropy of 56 bits.
- In general, the greater the entropy, the harder it is to break a cryptosystem.

Unicity Distance

- For a message of length *n*, the number of different keys that will decipher a ciphertext message to some intelligible plaintext in the same language as the original plaintext (such as an English text string) is given by the following formula:

$$2^{H(K)-nD} - 1$$

- Shannon defined the unicity distance, *U*, as an approximation of the amount of ciphertext such that the sum of the real information (entropy) in the corresponding plaintext plus the entropy of the encryption key equals the number of ciphertext bits used.

Unicity Distance

- Shannon then went on to show that ciphertexts longer than this distance are reasonably certain to have only one meaningful decryption.

- Ciphertexts significantly shorter than this are likely to have multiple, equally valid decryptions and therefore gain from the opponent's difficulty in choosing the correct one.

Unicity Distance

- For most symmetric cryptosystems, the unicity distance is defined as the entropy of the cryptosystem divided by the redundancy of the language.
  - $U = \mathrm{H}(K)/D$
- Unicity distance does not make deterministic predictions, but gives probabilistic results.
- Unicity distance estimates the minimum amount of ciphertext for which it is likely that there is only a single intelligible plaintext decryption when a brute-force attack is attempted.
- Generally, the longer the unicity distance, the better the cryptosystem. For DES, with a 56-bit key, and an ASCII English message, the unicity distance is about 8.2 ASCII characters or 66 bits.

Unicity Distance for ASCII

Table 11.1
Unicity Distances of ASCII Text Encrypted with Algorithms with Varying Key Lengths

Key Length (in bits)	Unicity Distance (in characters)
40	5.9
56	8.2
64	9.4
80	11.8
128	18.8
256	37.6

Unicity distance

- Unicity distance is not a measure of how much ciphertext is required for cryptanalysis, but how much ciphertext is required for there to be only one reasonable solution for cryptanalysis.

- The unicity distance is inversely proportional to the redundancy.
  - As redundancy approaches zero, even a trivial cipher can be unbreakable with a ciphertext-only attack.

Unicity distance and ideal secrecy

- Shannon defined a cryptosystem whose unicity distance is infinite as one that has ideal secrecy.

- If a cryptosystem has ideal secrecy, even successful cryptanalysis will leave some uncertainty about whether the recovered plaintext is the real plaintext.

Confusion and Diffusion

- The two basic techniques for obscuring the redundancies in a plaintext message are, according to Shannon, confusion and diffusion.
- Confusion obscures the relationship between the plaintext and the ciphertext. This frustrates attempts to study the ciphertext looking for redundancies and statistical patterns.
  - The easiest way to do this is through substitution. A simple substitution cipher, like the Caesar Cipher, is one in which every identical letter of plaintext is substituted for a single letter of ciphertext.
- Modern substitution ciphers are more complex: A long block of plaintext is substituted for a different block of ciphertext, and the mechanics of the substitution change with each bit in the plaintext or key.
  - This type of substitution is not necessarily enough; the German Enigma is a complex substitution algorithm that was broken during World War II.

Diffusion

- Diffusion dissipates the redundancy of the plaintext by spreading it out over the ciphertext.
- A cryptanalyst looking for those redundancies will have a harder time finding them. The simplest way to cause diffusion is through transposition (also called permutation).
- A simple transposition cipher, like columnar transposition, simply rearranges the letters of the plaintext.
- Modern ciphers do this type of permutation, but they also employ other forms of diffusion that can diffuse parts of the message throughout the entire message.

Outline

Mathematical Background

- Information Theory
- Complexity Theory
- Number Theory
- Factoring
- Prime Number Generation

Complexity Theory

- Complexity theory provides a methodology for analyzing the computational complexity of different cryptographic techniques and algorithms.
  - It compares cryptographic algorithms and techniques and determines their security.
- Information theory tells us that all cryptographic algorithms (except one-time pads) can be broken.
- Complexity theory tells us whether they can be broken before the heat death of the universe.

Complexity of Algorithms

- An algorithm's complexity is determined by the computational power needed to execute it.
- The computational complexity of an algorithm is often measured by two variables: *T* (for **time complexity**) and *S* (for **space complexity**, or memory requirement).
- Both *T* and *S* are commonly expressed as functions of *n*, where *n* is the size of the input.

Big O Notation

- Generally, the computational complexity of an algorithm is expressed in what is called “big O” notation: the order of magnitude of the computational complexity.
- It’s just the term of the complexity function which grows the fastest as n gets larger; all lower-order terms are ignored.
- For example, if the time complexity of a given algorithm is $4n^2 + 7n + 12$, then the computational complexity is on the order of n^2, expressed $O(n^2)$.

Polinomial-timeAlgorithms

- Generally, algorithms are classified according to their time or space complexities.
- An algorithm is **constant** if its complexity is independent of *n:* O(1).
- An algorithm is **linear**, if its time complexity is O(*n*).
- Algorithms can also be **quadratic**, **cubic**, and so on.
- All these algorithms are **polynomial**; their complexity is O(n^m), when *m* is a constant.
- The class of algorithms that have a polynomial time complexity are called **polynomial-time** algorithms.

Exponential Algorithms

- Algorithms whose complexities are $O(t^{f(n)})$, where t is a constant greater than 1 and f(n) is some polynomial function of n, are called exponential.
- The subset of exponential algorithms whose complexities are $O(c^{f(n)})$, where c is a constant and f(n) is more than constant but less than linear, is called superpolynomial.

Complexity and Cryptosystems

- Ideally, a cryptographer would like to be able to say that the best algorithm to break this encryption algorithm is of exponential-time complexity.
- In practice, the strongest statements that can be made, given the current state of the art of computational complexity theory, are of the form "all known cracking algorithms for this cryptosystem are of superpolynomial-time complexity."
- That is, the cracking algorithms that we know are of superpolynomial-time complexity, but it is not yet possible to prove that no polynomial-time cracking algorithm could ever be discovered.
- Advances in computational complexity may some day make it possible to design algorithms for which the existence of polynomial-time cracking algorithms can be ruled out with mathematical certainty.

Classes of Algorithms

Table 11.2
Running Times of Different Classes of Algorithms

Class	Complexity	# of Operations for $n = 10^6$	Time at 10^6 O/S
Constant	O(1)	1	1 μsec.
Linear	O(n)	10^6	1 sec.
Quadratic	O(n^2)	10^{12}	11.6 days
Cubic	O(n^3)	10^{18}	32, 000 yrs.
Exponential	O(2^n)	$10^{301,030}$	$10^{301,006}$ times the age of the universe

Complexity of Problems

- Complexity theory also classifies the inherent complexity of problems, not just the complexity of particular algorithms used to solve problems.
- The theory looks at the minimum time and space required to solve the hardest instance of a problem on a theoretical computer known as a Turing machine.
- A Turing machine is a finite-state machine with an infinite read-write memory tape.
- It turns out that a Turing machine is a realistic model of computation.

Tractable and Intractable Problems

- Problems that can be solved with polynomial-time algorithms are called tractable, because they can usually be solved in a reasonable amount of time for reasonable-sized inputs.
- Problems that cannot be solved in polynomial time are called intractable, because calculating their solution quickly becomes infeasible.
- Intractable problems are sometimes just called hard.
- Problems that can only be solved with algorithms that are superpolynomial are computationally intractable, even for relatively small values of *n*.

Undecidable problems and classes of Complexity

- It gets worse. Alan Turing proved that some problems are undecidable.

  It is impossible to devise any algorithm to solve them, regardless of the algorithm's time complexity.

- **P** consists of all problems that can be solved in polynomial time.
- The class **NP** consists of all problems that can be solved in polynomial time only on a nondeterministic Turing machine: a variant of a normal Turing machine that can make guesses.
- The machine guesses the solution to the problem — either by making "lucky guesses" or by trying all guesses in parallel — and checks its guess in polynomial time.

NP and Cryptography

- **NP** ’s relevance to cryptography is this: Many symmetric algorithms and all public-key algorithms can be cracked in nondeterministic polynomial time.
- Given a ciphertext *C*, the cryptanalyst simply guesses a plaintext, *X*, and a key, *k*, and in polynomial time runs the encryption algorithm on inputs *X* and *k* and checks whether the result is equal to *C*.
- This is important theoretically, because it puts an upper bound on the complexity of cryptanalysis for these algorithms.
- In practice, of course, it is a deterministic polynomial-time algorithm that the cryptanalyst seeks.
- Furthermore, this argument is not applicable to all classes of ciphers; in particular, it is not applicable to one-time pads.

NP-complete problems

- If all **NP** problems are solvable in polynomial time on a deterministic machine, then **P = NP**.
- Although it seems obvious that some **NP** problems are much harder than others (a brute-force attack against an encryption algorithm versus encrypting a random block of plaintext), it has never been proven that **P ≠ NP** (or that **P = NP**).
- However, most people working in complexity theory believe that they are unequal.

- NP-complete problem: if this problem can be solved in polynomial time, then P = NP.
  - Satisfiability problem (given a propositional Boolean formula, is there a way to assign truth values to the variables that makes the formula true?) is **NP-complete**

NP-Complete Problems

- Michael Garey and David Johnson compiled a list of over 300 **NP-complete problems**. Here are just a few of them:
- Traveling Salesman Problem. A traveling salesman has to visit *n* different cities using only one tank of gas (there is a maximum distance he can travel).
  - Is there a route that allows him to visit each city exactly once on that single tank of gas?

Outline

Mathematical Background

- Information Theory
- Complexity Theory
- Number Theory
- Factoring
- Prime Number Generation

Generators

- If p is a prime, and g is less than p, then g is a **generator** mod p if
  - for each b from 1 to p - 1, there exists some a where $g^a \equiv b \pmod{p}$.
- Another way of saying this is that g is **primitive** with respect to p.
- *For example, if p = 11, 2 is a generator mod 11:*
  - $2^{10} = 1024 \equiv 1 \pmod{11}$
  - $2^1 = 2 \equiv 2 \pmod{11}$
  - $2^8 = 256 \equiv 3 \pmod{11}$
  - $2^2 = 4 \equiv 4 \pmod{11}$
  - $2^4 = 16 \equiv 5 \pmod{11}$
  - $2^9 = 512 \equiv 6 \pmod{11}$
  - $2^7 = 128 \equiv 7 \pmod{11}$
  - $2^3 = 8 \equiv 8 \pmod{11}$
  - $2^6 = 64 \equiv 9 \pmod{11}$
  - $2^5 = 32 \equiv 10 \pmod{11}$
- Every number from 1 to 10 can be expressed as $2^a \pmod{p}$.

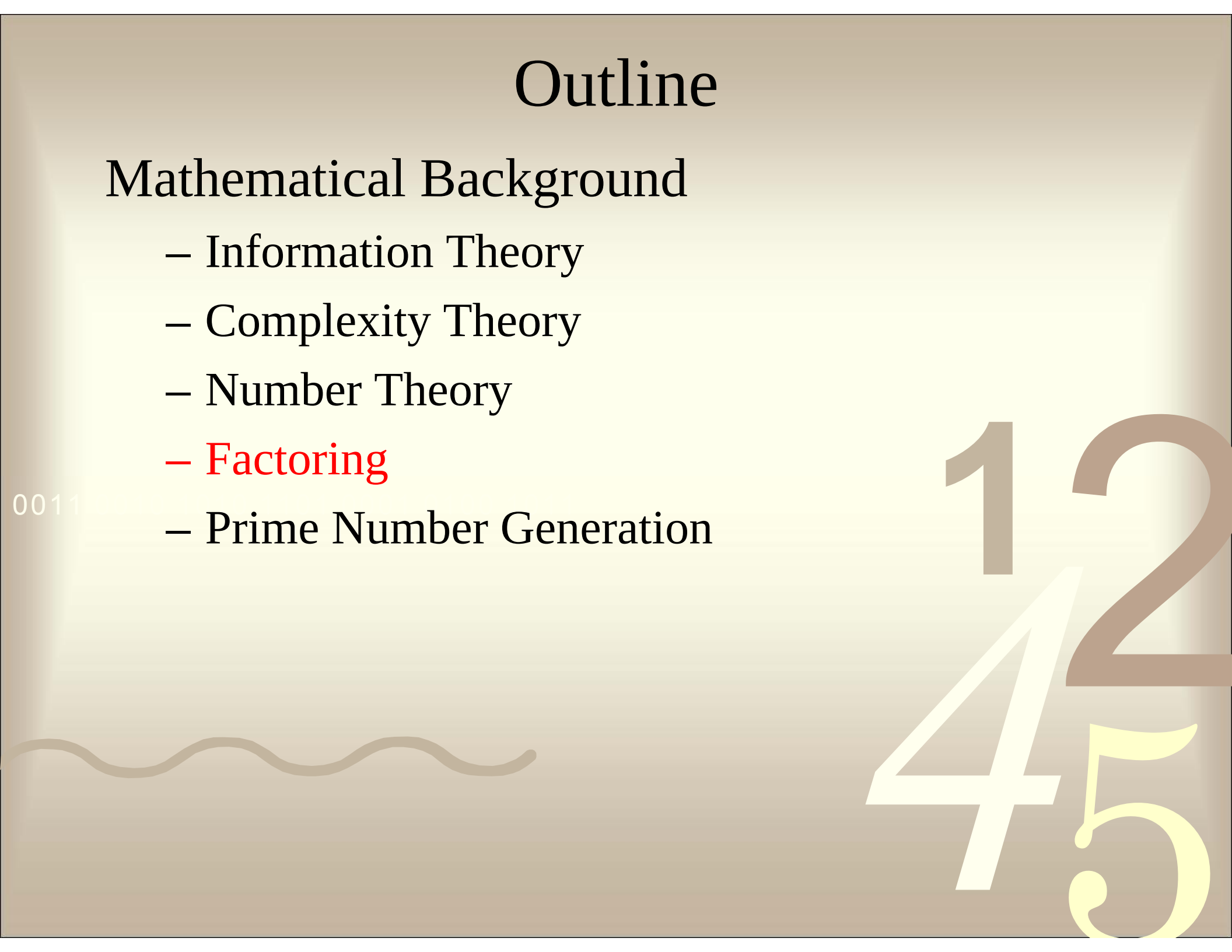
Generators

- For $p = 11$, the generators are 2, 6, 7, and 8. The other numbers are not generators.
- For example, 3 is not a generator because there is no solution to
  - $3^a = 2 \pmod{11}$
- In general, testing whether a given number is a generator is not an easy problem.

Outline

Mathematical Background

- Information Theory
- Complexity Theory
- Number Theory
- Factoring
- Prime Number Generation



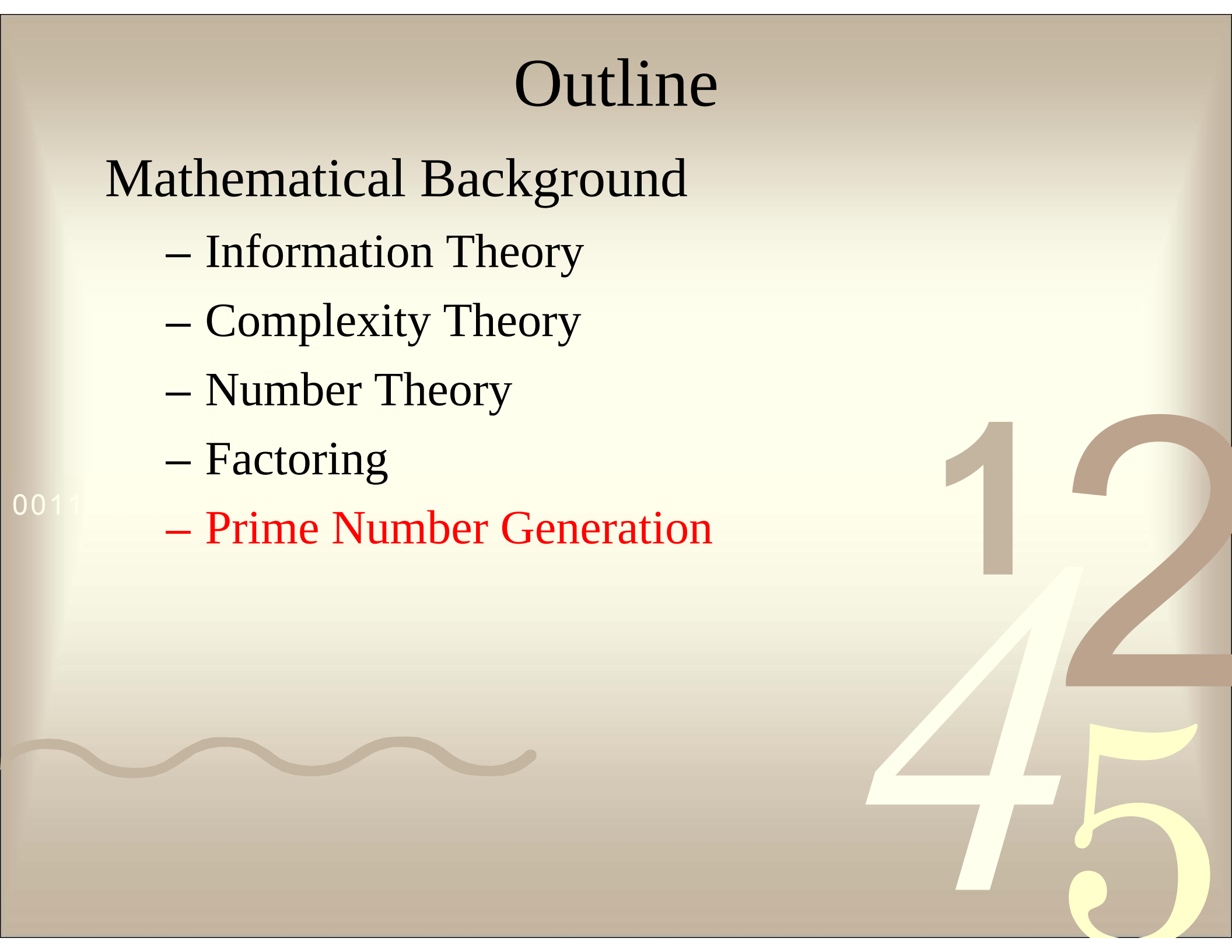
Factoring

- Factoring a number means finding its prime factors.
- Quadratic sieve (QS). This is the fastest-known algorithm for numbers less than 110 decimal digits long and has been used extensively.
  - A faster version of this algorithm is called the multiple polynomial quadratic sieve. The fastest version of this algorithm is called the double large prime variation of the multiple polynomial quadratic sieve.
- Elliptic curve method (ECM). This method has been used to find 43-digit factors, but nothing larger.
- Pollard's Monte Carlo algorithm.
- Continued fraction algorithm.
- Trial division.

Outline

Mathematical Background

- Information Theory
- Complexity Theory
- Number Theory
- Factoring
- Prime Number Generation



Prime Number Generation

- Public-key algorithms need prime numbers. Before discussing the mathematics of prime number generation, let's answer a few obvious questions.
  1. If everyone needs a different prime number, won't we run out?
     - No. In fact, there are approximately 10^{151} primes 512 bits in length or less.
     - There are only 10^{77} atoms in the universe. If every atom in the universe needed a billion new primes every microsecond from the beginning of time until now, you would only need 10^{109} primes; there would still be approximately 10^{151} 512-bit primes left.

Prime Number Generation

2. What if two people accidentally pick the same prime number?

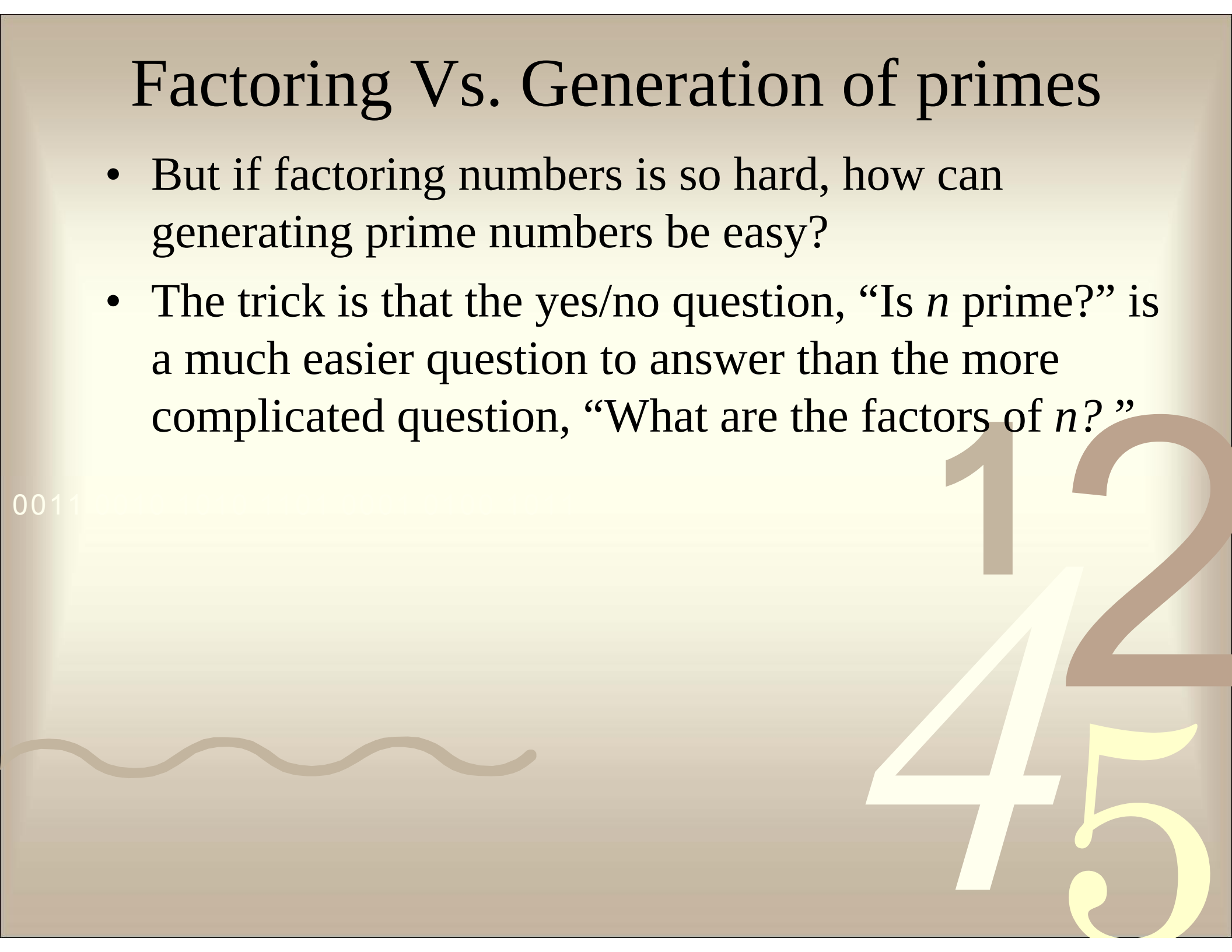
– It won't happen. With over 10^{151} prime numbers to choose from, the odds of that happening are significantly low.

3. If someone creates a database of all primes, won't he be able to use that database to break public-key algorithms?

- Yes, but he can't do it. If you could store one gigabyte of information on a drive weighing one gram, then a list of just the 512-bit primes would weigh so much that it would exceed the Chandrasekhar limit and collapse into a black hole...so you couldn't retrieve the data anyway.

Factoring Vs. Generation of primes

- But if factoring numbers is so hard, how can generating prime numbers be easy?
- The trick is that the yes/no question, “Is *n* prime?” is a much easier question to answer than the more complicated question, “What are the factors of *n?* ”



Generating Primes

- The wrong way to find primes is to generate random numbers and then try to factor them.
- The right way is to generate random numbers and test if they are prime.
- There are several probabilistic primality tests: tests that determine whether a number is prime with a given degree of confidence.
- Assuming this “degree of confidence” is large enough, these sorts of tests are good enough.

Solovay-Strassen

- Their algorithm uses the Jacobi symbol to test if p is prime:
  - **(1)** Choose a random number, a, less than p.
  - **(2)** If the gcd(a,p) ≠ 1, then p fails the test and is composite.
  - **(3)** Calculate j = a(p- 1)/2 mod p.
  - **(4)** Calculate the Jacobi symbol J(a,p).
  - **(5)** If j ≠ J(a,p), then p is definitely not prime.
  - **(6)** If j = J(a,p), then the likelihood that p is not prime is no more than 50 percent.

Lehmann

- Another, simpler, test was developed independently by Lehmann. Here it tests if p is prime:
  - **(1)** Choose a random number a less than p.
  - **(2)** Calculate $a^{(p-1)/2} \bmod p$.
  - **(3)** If $a^{(p-1)/2} \not\equiv 1$ or $-1 \pmod{p}$, then p is definitely not prime.
  - **(4)** If $a^{(p-1)/2} \equiv 1$ or $-1 \pmod{p}$, then the likelihood that p is not prime is no more than 50 percent.

Rabin-Miller

- The algorithm everyone uses—it's easy—was developed by Michael Rabin.
- Choose a random number, *p*, to test. Calculate *b*, where *b* is the number of times 2 divides *p* - 1 (i.e., 2b is the largest power of 2 that divides *p* - 1). Then calculate *m*, such that *p* = 1 + 2b **m*.
  - **(1)** Choose a random number, *a*, such that *a* is less than *p*.
  - **(2)** Set *j* = 0 and set *z* = am mod *p*.
  - **(3)** If *z* = 1, or if *z* = p - 1, then *p* passes the test and may be prime.
  - **(4)** If *j* > 0 and *z* = 1, then *p* is not prime.
  - **(5)** Set *j* = j + 1. If *j* < b and *z* ≠ *p* - 1, set *z* = z2 mod *p* and go back to step (4). If *z* = p - 1, then *p* passes the test and may be prime.
  - **(6)** If *j* = b and *z* ≠ *p* - 1, then *p* is not prime.

Practical Considerations

- In real-world implementations, prime generation goes quickly.
  - **(1)** Generate a random *n*- bit number, *p*.
  - **(2)** Set the high-order and low-order bit to 1. (The high-order bit ensures that the prime is of the required length and the low-order bit ensures that it is odd.)
  - **(3)** Check to make sure *p* is not divisible by any small primes: 3, 5, 7, 11, and so on. Many implementations test *p* for divisibility by all primes less than 256. The most efficient is to test for divisibility by all primes less than 2000.
  - **(4)** Perform the Rabin-Miller test for some random *a*. If *p* passes, generate another random *a* and go through the test again. Choose a small value of *a* to make the calculations go quicker. Do five tests.
  - If *p* fails one of the tests, generate another *p* and try again.

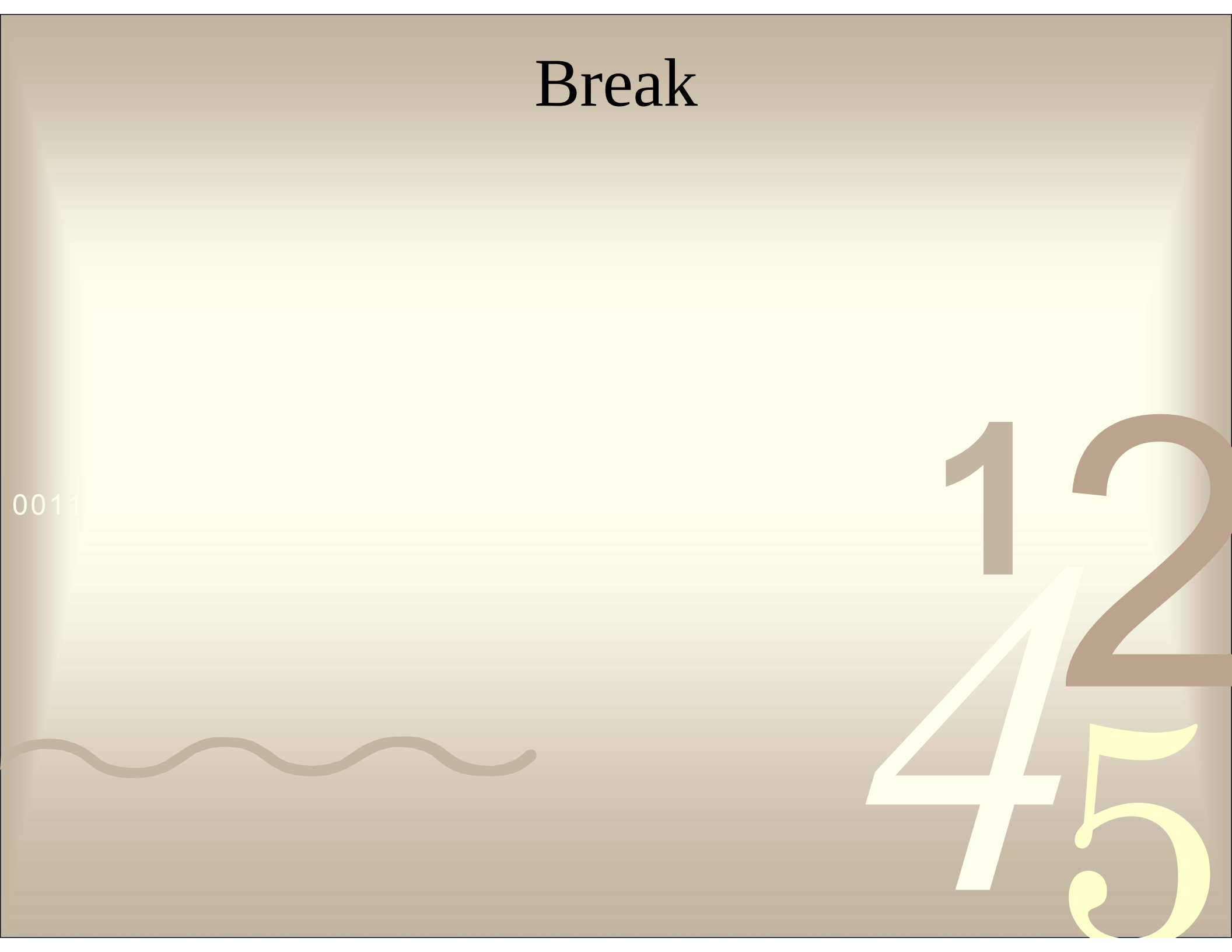
Strong Primes

- If n is the product of two primes, p and q, it may be desirable to use strong primes for p and q. These are prime numbers with certain properties that make the product n difficult to factor by specific factoring methods. Among the properties suggested have been:
  - The greatest common divisor of $p - 1$ and $q - 1$ should be small.
  - Both $p - 1$ and $q - 1$ should have large prime factors, respectively p' and q'.
  - Both $p' - 1$ and $q' - 1$ should have large prime factors.
  - Both $p + 1$ and $q + 1$ should have large prime factors.
  - Both $(p - 1)/2$ and $(q - 1)/2$ should be prime.

Strong Primes

- Whether strong primes are necessary is a subject of debate.
- These properties were designed to fight some older factoring algorithms. However, the fastest factoring algorithms have as good a chance of factoring numbers that meet these criteria as they do of factoring numbers that do not.
- Schneier recommends against specifically generating strong primes.
- The length of the primes is much more important than the structure. Moreover, structure may be damaging because it is less random.

Break



Outline

- Data Encryption Standard (DES)
  - Background
  - Description of DES
  - Security of DES
  - Differential and Linear Cryptanalysis
  - The Real Design Criteria
  - DES Variants
  - How Secure Is DES Today?

Background

- The Data Encryption Standard (DES), known as the Data Encryption Algorithm (DEA) by ANSI and the DEA-1 by the ISO, was a worldwide standard for many years.
- Although now it is not safe any more, it held up remarkably well against years of cryptanalysis and was secure against all but possibly the most powerful of adversaries until 2000s.

Development of the Standard

- In the early 1970s, nonmilitary cryptographic research was haphazard. Almost no research papers were published in the field.
- Most people knew that the military used special coding equipment to communicate, but few understood the science of cryptography.
- The National Security Agency (NSA) had considerable knowledge, but they did not even publicly admit their own existence.
- Buyers didn't know what they were buying. Several small companies made and sold cryptographic equipment, primarily to overseas governments.
- The equipment was all different and couldn't interoperate.
- No one really knew if any of it was secure; there was no independent body to certify the security.

NIST

- In 1972, the National Bureau of Standards (NBS), now the National Institute of Standards and Technology (NIST), initiated a program to protect computer and communications data.
- As part of that program, they wanted to develop a single, standard cryptographic algorithm.
- A single algorithm could be tested and certified, and different cryptographic equipment using it could interoperate.
- It would also be cheaper to implement and readily available.

Proposal for Standards

- In the May 15, 1973 *Federal Register*, the NBS issued a public request for proposals for a standard cryptographic algorithm. They specified a series of design criteria:
  - The algorithm must provide a high level of security.
  - The algorithm must be completely specified and easy to understand.
  - The security of the algorithm must reside in the key; the security should not depend on the secrecy of the algorithm.
  - The algorithm must be available to all users.
  - The algorithm must be adaptable for use in diverse applications.
  - The algorithm must be economically implementable in electronic devices.
  - The algorithm must be efficient to use.
  - The algorithm must be able to be validated.
  - The algorithm must be exportable.

Lucifer

- The NBS issued a second request in the August 27, 1974 *Federal Register*.
- Eventually they received a promising candidate: an algorithm based on one developed by IBM during the early 1970s, called Lucifer.
- The algorithm, although complicated, was straightforward. It used only simple logical operations on small groups of bits and could be implemented fairly efficiently in hardware.
- The NBS requested the NSA's help in evaluating the algorithm's security and determining its suitability as a federal standard.

NSA role

- Finally, in the March 17, 1975 *Federal Register*, the NBS published both the details of the algorithm and IBM's statement granting a nonexclusive, royalty-free license for the algorithm, and requested comment.
- And there were comments.
  - Many were wary of the NSA's "invisible hand" in the development of the algorithm.
  - They were afraid that the NSA had modified the algorithm to install a trapdoor. They complained that the NSA reduced the key size from the original 128-bits to 56-bits..
  - They also complained about the inner workings of the algorithm.

Finally DES

- In 1976, the NBS held two workshops to evaluate the proposed standard.
- The first workshop discussed the mathematics of the algorithm and the possibility of a trapdoor .
- The second workshop discussed the possibility of increasing the algorithm's key length. The algorithm's designers, evaluators, implementors, vendors, users, and critics were invited.
- Despite criticism, the Data Encryption Standard was adopted as a federal standard on November 23, 1976 and authorized for use on all unclassified government communications.
- The official description of the standard, FIPS PUB 46, "Data Encryption Standard," was published on January 15, 1977 and became effective six months later. FIPS PUB 81, "DES Modes of Operation," was published in 1980. FIPS PUB 74, "Guidelines for Implementing and Using the NBS Data Encryption Standard," was published in 1981. NBS also published FIPS PUB 112, specifying DES for password encryption, and FIPS PUB 113, specifying DES for computer data authentication. (FIPS stands for Federal Information Processing Standard.)

NSA-NBS Misunderstanding

- These standards were unprecedented. Never before had an NSA-evaluated algorithm been made public.
- This was probably the result of a misunderstanding between NSA and NBS. The NSA thought DES was hardware-only. The standard mandated a hardware implementation, but NBS published enough details so that people could write DES software.
- Off the record, NSA has characterized DES as one of their biggest mistakes.
  - If they knew the details would be released so that people could write software, they would never have agreed to it. DES did more to galvanize the field of cryptanalysis than anything else.
- Now there was an algorithm to study: one that the NSA said was secure.
- It is no accident that the next government standard algorithm, Skipjack was classified.

Adoption of the Standard

- The American National Standards Institute (ANSI) approved DES as a private-sector standard in 1981 (ANSI X3.92).They called it the Data Encryption Algorithm (DEA). ANSI published a standard for DEA modes of operation (ANSI X3.106).
- Two other groups within ANSI, representing retail and wholesale banking, developed DES-based standards. Retail banking involves transactions between financial institutions and private individuals, and wholesale banking involves transactions between financial institutions.
- ANSI's Financial Institution Retail Security Working Group developed a standard for the management and security of PINs (ANSI X9.8) and another DES-based standard for the authentication of retail financial messages (ANSI X9.19). The group has a draft standard for secure key distribution (ANSI X9.24).
- ANSI's Financial Institution Wholesale Security Working Group developed its own set of standards for message authentication (ANSI X9.9), key management (ANSI X9.17), encryption (ANSI X9.23), and secure personal and node authentication (ANSI X9.26).
- The American Bankers Association develops voluntary standards for the financial industry. They published a standard recommending DES for encryption, and another standard for managing cryptographic keys.

Validation and Certification of DES Equipment

- As part of the DES standard, NIST validates implementations of DES.
- This validation confirms that the implementation follows the standard.
- Until 1994, NIST only validated hardware and firmware implementations — until then the standard prohibited software implementations. As of March 1995, 73 different implementations had been validated.
- NIST also developed a program to certify that authentication equipment conformed to ANSI X9.9 and FIPS 113.
- As of March, 1995, 33 products had been validated.
- The Department of the Treasury has an additional certification procedure.
- NIST also has a program to confirm that equipment conforms to ANSI X9.17 for wholesale key management; four products have been validated as of March, 1995.

Outline

- Data Encryption Standard (DES)
  - Background
  - Description of DES
  - Security of DES
  - Differential and Linear Cryptanalysis
  - The Real Design Criteria
  - DES Variants
  - How Secure Is DES Today?

DES: a block cipher

- DES is a block cipher; it encrypts data in 64-bit blocks.
- DES is a symmetric algorithm: The same algorithm and key are used for both encryption and decryption (except for minor differences in the key schedule).
- The key length is 56 bits. (The key is usually expressed as a 64-bit number, but every eighth bit is used for parity checking and is ignored. These parity bits are the least-significant bits of the key bytes.)
  - The key can be any 56-bit number and can be changed at any time.
  - A handful of numbers are considered weak keys, but they can easily be avoided.
  - All security rests within the key.

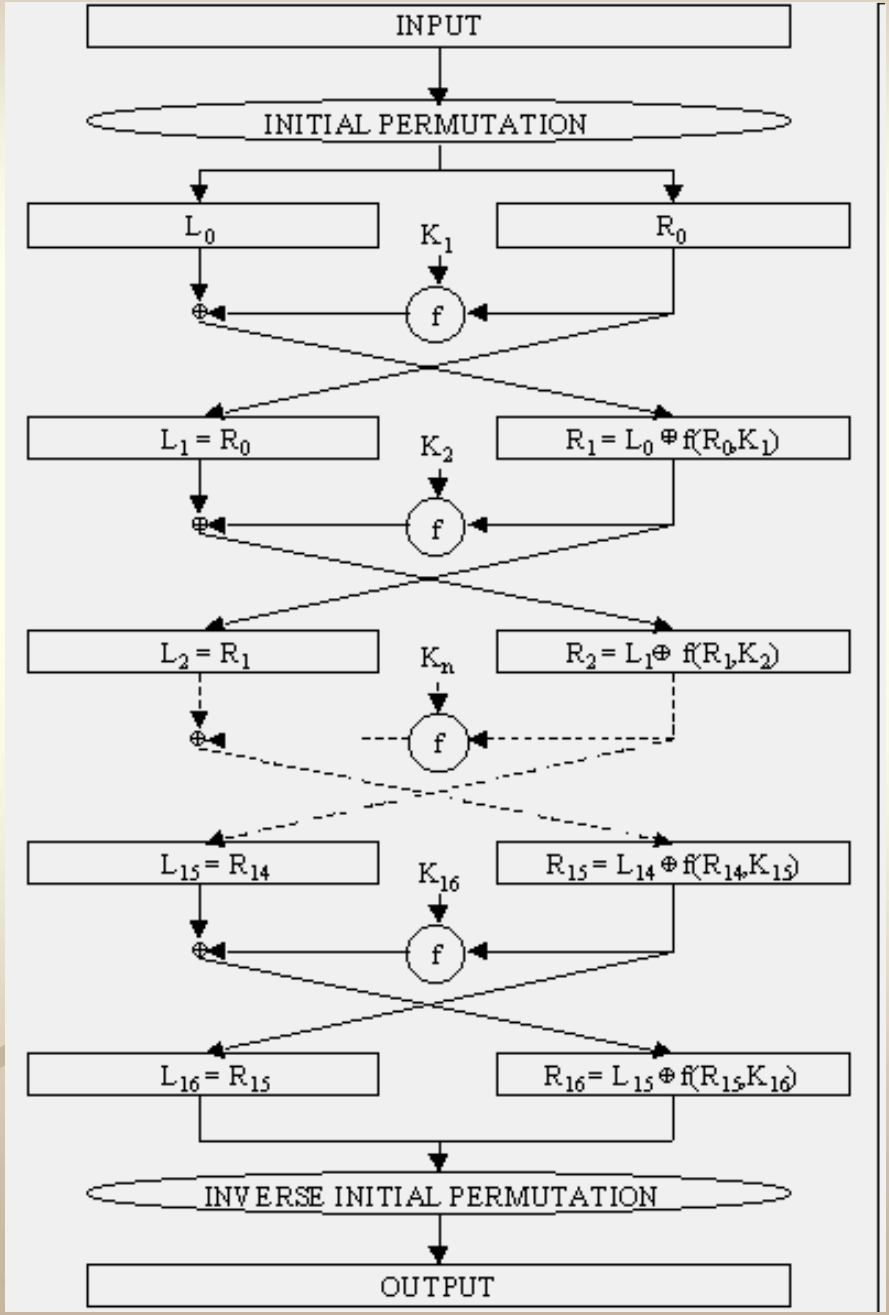
DES: confusion and diffusion

- At its simplest level, the algorithm is nothing more than a combination of the two basic techniques of encryption: confusion and diffusion.
- The fundamental building block of DES is a single combination of these techniques (a substitution followed by a permutation) on the text, based on the key.
- This is known as a round. DES has 16 rounds; it applies the same combination of techniques on the plaintext block 16 times.
- The algorithm uses only standard arithmetic and logical operations on numbers of 64 bits at most, so it was easily implemented in late 1970s hardware technology.
- The repetitive nature of the algorithm makes it ideal for use on a special-purpose chip.
- Initial software implementations were clumsy, but current implementations are better.

Outline of the Algorithm

- DES operates on a 64-bit block of plaintext.
- After an initial permutation, the block is broken into a right half and a left half, each 32 bits long.
- Then there are 16 rounds of identical operations, called *Function f*, in which the data are combined with the key.
- After the sixteenth round, the right and left halves are joined, and a final permutation (the inverse of the initial permutation) finishes off the algorithm.

DES: 16 rounds



Outline of the Algorithm

- In each round, the key bits are shifted, and then 48 bits are selected from the 56 bits of the key.
- The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again.
- These four operations make up Function f. The output of Function f is then combined with the left half via another XOR.
- The result of these operations becomes the new right half; the old right half becomes the new left half. These operations are repeated 16 times, making 16 rounds of DES.

Single round of DES

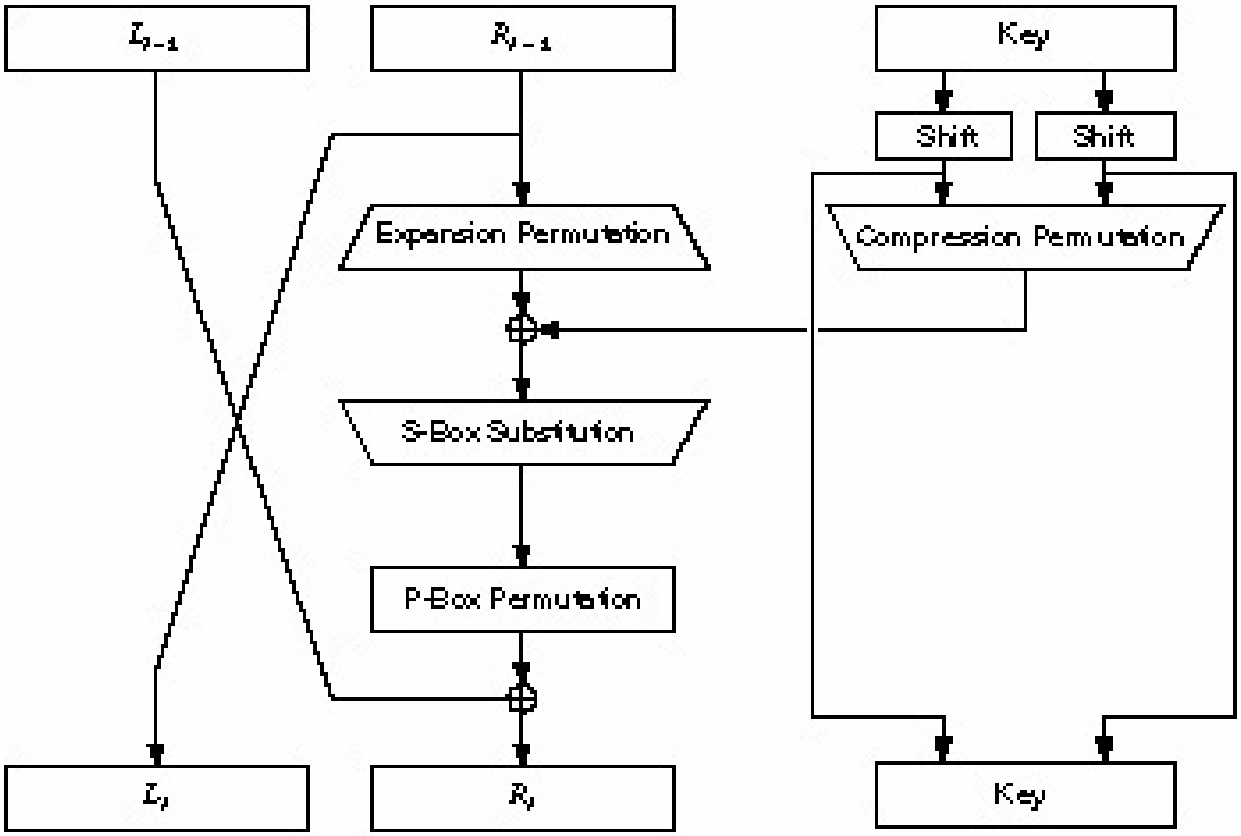


Figure 12.2 One round of DES.

The Initial Permutation

- The initial permutation occurs before round 1; it transposes the input blocks.
  - For example, the initial permutation moves bit 58 of the plaintext to bit position 1, bit 50 to bit position 2, bit 42 to bit position 3, and so forth.
- Since this bit-wise permutation is difficult in software (although it is trivial in hardware), many software implementations of DES leave out both the initial and final permutations.
  - While this new algorithm is no less secure than DES, it does not follow the DES standard and should not be called DES.

Initial Permutation

Table 12.1
Initial Permutation

58,	50,	42,	34,	26,	18,	10,	2,	60,	52,	44,	36,	28,	20,	12,	4,
62,	54,	46,	38,	30,	22,	14,	6,	64,	56,	48,	40,	32,	24,	16,	8,
57,	49,	41,	33,	25,	17,	9,	1,	59,	51,	43,	35,	27,	19,	11,	3,
61,	53,	45,	37,	29,	21,	13,	5,	63,	55,	47,	39,	31,	23,	15,	7

The Key Transformation

- Initially, the 64-bit DES key is reduced to a 56-bit key by ignoring every eighth bit.
- These bits can be used as parity check to ensure the key is error-free.
- After the 56-bit key is extracted, a different 48-bit subkey is generated for each of the 16 rounds of DES.
- These subkeys, K_i are determined in the following manner.
- First, the 56-bit key is divided into two 28-bit halves. Then, the halves are circularly shifted left by either one or two bits, depending on the round.

Key permutation and shifting

Table 12.2
Key Permutation

57,	49,	41,	33,	25,	17,	9,	1,	58,	50,	42,	34,	26,	18,
10,	2,	59,	51,	43,	35,	27,	19,	11,	3,	60,	52,	44,	36,
63,	55,	47,	39,	31,	23,	15,	7,	62,	54,	46,	38,	30,	22,
14,	6,	61,	53,	45,	37,	29,	21,	13,	5,	28,	20,	12,	4

Table 12.3
Number of Key Bits Shifted per Round

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Compression permutation

- After being shifted, 48 out of the 56 bits are selected. Because this operation permutes the order of the bits as well as selects a subset of bits, it is called a compression permutation.

- This operation provides a subset of 48 bits.

- Because of the shifting, a different subset of key bits is used in each subkey. Each bit is used in approximately 14 of the 16 subkeys, although not all bits are used exactly the same number of times.

Compression permutation

Table 12.4
Compression Permutation

14,	17,	11,	24,	1,	5,	3,	28,	15,	6,	21,	10,
23,	19,	12,	4,	26,	8,	16,	7,	27,	20,	13,	2,
41,	52,	31,	37,	47,	55,	30,	40,	51,	45,	33,	48,
44,	49,	39,	56,	34,	53,	46,	42,	50,	36,	29,	32

- For example, the bit in position 33 of the shifted key moves to position 35 of the output.

The Expansion Permutation

- This operation expands the right half of the data, R_i, from 32 bits to 48 bits.
- Because this operation changes the order of the bits as well as repeating certain bits, it is known as an expansion permutation.
- This operation has two purposes:
  - It makes the right half the same size as the key for the XOR operation and it provides a longer result that can be compressed during the substitution operation.

The S-Box Substitution

- After the compressed key is XORed with the expanded block, the 48-bit result moves to a substitution operation.
- The substitutions are performed by eight substitution boxes, or S-boxes.
- Each S-box has a 6-bit input and a 4-bit output, and there are eight different S-boxes.
- The 48 bits are divided into eight 6-bit sub-blocks. Each separate block is operated on by a separate S-box: The first block is operated on by S-box 1, the second block is operated on by S-box 2, and so on.

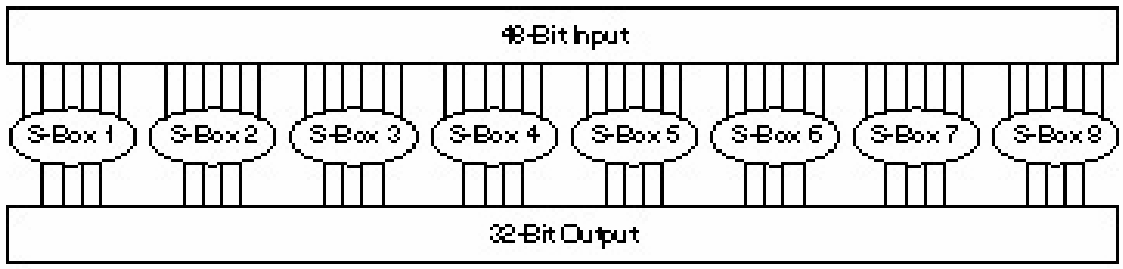


Figure 12.4 S-box substitution.

S-Box structure

- Each S-box is a table of 4 rows and 16 columns. Each entry in the box is a 4-bit number. The 6 input bits of the S-box specify under which row and column number to look for the output.

Table 12.6
S-Boxes

S-box 1:

14,	4,	13,	1,	2,	15,	11,	8,	3,	10,	6,	12,	5,	9,	0,	7,
0,	15,	7,	4,	14,	2,	13,	1,	10,	6,	12,	11,	9,	5,	3,	8,
4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0,
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13,

- The input bits specify an entry in the S-box in a very particular manner. Consider an S-box input of 6 bits, labeled $b1$,$b2$,$b3$,$b4$,$b5$ and $b6$.
- Bits $b1$ and $b6$ are combined to form a 2-bit number, from 0 to 3, which corresponds to a row in the table. The middle 4 bits, $b2$ through $b5$ are combined to form a 4-bit number, from 0 to 15, which corresponds to a column in the table.

The P-Box Permutation

- The 32-bit output of the S-box substitution is permuted according to a P-box.
- This permutation maps each input bit to an output position; no bits are used twice and no bits are ignored. This is called a straight permutation or just a permutation.
- Table 12.7 shows the position to which each bit moves. For example, bit 21 moves to bit 4, while bit 4 moves to bit 31.

Table 12.7
P-Box Permutation

16,	7,	20,	21,	29,	12,	28,	17,	1,	15,	23,	26,	5,	18,	31,	10,
2,	8,	24,	14,	32,	27,	3,	9,	19,	13,	30,	6,	22,	11,	4,	25

The Final Permutation

- The final permutation is the inverse of the initial permutation.

Table 12.8
Final Permutation

40,	8,	48,	16,	56,	24,	64,	32,	39,	7,	47,	15,	55,	23,	63,	31,
38,	6,	46,	14,	54,	22,	62,	30,	37,	5,	45,	13,	53,	21,	61,	29,
36,	4,	44,	12,	52,	20,	60,	28,	35,	3,	43,	11,	51,	19,	59,	27,
34,	2,	42,	10,	50,	18,	58,	26,	33,	1,	41,	9,	49,	17,	57,	25

Decrypting DES

- After all the substitutions, permutations, XORs, and shifting around, you might think that the decryption algorithm is completely different and just as confusing as the encryption algorithm.

- On the contrary, the various operations were chosen to produce a very useful property: The same algorithm works for both encryption and decryption.

Decrypting DES

- With DES it is possible to use the same function to encrypt or decrypt a block. The only difference is that the keys must be used in the reverse order.
- That is, if the encryption keys for each round are K_1 K_2 K_3,..., K_{16} then the decryption keys are K_{16} K_{15} K_{14}, ..., K_1.
- The algorithm that generates the key used for each round is circular as well.
- The key shift is a right shift and the number of positions shifted is 0,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1.

Modes of DES

- FIPS PUB 81 specifies four modes of operation: ECB, CBC, OFB, and CFB.
- The ANSI banking standards specify ECB and CBC for encryption, and CBC and *n*-bit CFB for authentication.

Outline

- Data Encryption Standard (DES)
  - Background
  - Description of DES
  - Security of DES
  - Differential and Linear Cryptanalysis
  - The Real Design Criteria
  - DES Variants
  - How Secure Is DES Today?

Security of DES

- People have long questioned the security of DES.
- There has been much speculation on the key length, number of iterations, and design of the S-boxes. The S-boxes were particularly mysterious — all those constants, without any apparent reason as to why or what they're for.
- Although IBM claimed that the inner workings were the result of 17 man-years of intensive cryptanalysis some people feared that the NSA embedded a trapdoor into the algorithm so they would have an easy means of decrypting messages.
- The U.S. Senate Select Committee on Intelligence, with full top-secret clearances, investigated the matter in 1978. The findings of the committee are classified, but an unclassified summary of those findings exonerated the NSA from any improper involvement in the algorithm's design.

Security of DES

- Although more information has been published on the cryptanalysis of DES than any other block cipher, the most practical attack is a brute force approach.
- Various minor cryptanalytic properties are known, and three theoretical attacks are possible which, while having a theoretical complexity less than a brute force attack, require an unrealistic amount of known or chosen plaintext to carry out.

Outline

- Data Encryption Standard (DES)
  - Background
  - Description of DES
  - Security of DES
  - Differential and Linear Cryptanalysis
  - The Real Design Criteria
  - DES Variants
  - How Secure Is DES Today?

Differential Cryptanalysis

- In 1990, Eli Biham and Adi Shamir introduced differential cryptanalysis.
- This is a new method of cryptanalysis, unknown before to the public. Using this method, Biham and Shamir found a chosen-plaintext attack against DES that was more efficient than brute force.
- Differential cryptanalysis looks specifically at ciphertext pairs: pairs of ciphertexts whose plaintexts have particular differences.
- It analyzes the evolution of these differences as the plaintexts propagate through the rounds of DES when they are encrypted with the same key.

Simplified Differential Cryptanalysis

- Simply, choose pairs of plaintexts with a fixed difference.
- The two plaintexts can be chosen at random, as long as they satisfy particular difference conditions; the cryptanalyst does not even have to know their values.
- (For DES, the term “difference” is defined using XOR. This can be different for different algorithms.)
- Then, using the differences in the resulting ciphertexts, assign different probabilities to different keys. As you analyze more and more ciphertext pairs, one key will emerge as the most probable. This is the correct key.

  (The details are more complicated!!!)

Complexity of Differential Cryptanalysis

Table 12.14
Differential Cryptanalysis Attacks against DES

No. of Rounds	Chosen Plaintexts	Known Plaintexts	Analyzed Plaintexts	Complexity of Analysis
8	2^{14}	2^{38}	4	2^{9}
9	2^{24}	2^{44}	2	$2^{32\dagger}$
10	2^{24}	2^{43}	2^{14}	2^{15}
11	2^{31}	2^{47}	2	$2^{32\dagger}$
12	2^{31}	2^{47}	2^{21}	2^{21}
13	2^{39}	2^{52}	2	$2^{32\dagger}$
14	2^{39}	2^{51}	2^{29}	2^{29}
15	2^{47}	2^{56}	2^{7}	2^{37}
16	2^{47}	2^{55}	2^{36}	2^{37}

Increasing DES Resistance

- The best attack against full 16-round DES requires 2^{47} chosen plaintexts. This can be converted to a known plaintext attack, but that requires 2^{55} known plaintexts. And 2^{37} DES operations are required during analysis.
- DES's resistance can be improved by increasing the number of rounds.
  - Chosen-plaintext differential cryptanalysis DES with 17 or 18 rounds takes about the same time as a brute-force search.
- At 19 rounds or more, differential cryptanalysis becomes impossible because it requires more than 2^{64} chosen plaintexts: Remember, DES has a 64-bit block size, so it only *has* 2^{64} possible plaintext blocks.
- *In general, you can prove that an algorithm is resistant to differential cryptanalysis by showing that the amount of plaintext required to mount such an attack is greater than the amount of plaintext possible.*

Differential: A Theoretical Attack

- First, this attack is largely theoretical.
- The enormous time and data requirements to mount a differential cryptanalytic attack put it beyond the reach of almost everyone.
- To get the requisite data for this attack against a full DES, you have to encrypt a 1.5 megabits-per-second data stream of *chosen plaintext* for almost three years.
- For full 16-round DES, the attack is slightly less efficient than brute force (the differential cryptanalytic attack requires $2^{55.1}$ operations, and brute force requires 2^{55}).
- The consensus (until DEA was introduced) was that DES, when implemented properly, was still secure against differential cryptanalysis.

Related-Key Cryptanalysis

- Related-key cryptanalysis is similar to differential cryptanalysis, but it examines the difference between keys.
- The attack is different from any previously discussed: The cryptanalyst chooses a relationship between a pair of keys, but does not know the keys themselves.
- Data is encrypted with both keys.
  - In the known-plaintext version, the cryptanalyst knows the plaintext and ciphertext of data encrypted with the two keys.
  - In the chosen-plaintext version, the cryptanalyst gets to choose the plaintext encrypted with the two keys.

Linear Cryptanalysis

- Linear cryptanalysis is another type of cryptanalytic attack, invented by Mitsuru Matsui.
- This attack uses linear approximations to describe the action of a block cipher (in this case, DES.)
- This means that if you XOR some of the plaintext bits together, XOR some ciphertext bits together, and then XOR the result, you will get a single bit that is the XOR of some of the key bits.
- This is a linear approximation and will hold with some probability p. If $p \neq ½$, then this bias can be exploited.
- Use collected plaintexts and associated ciphertexts to guess the values of the key bits.
- The more data you have, the more reliable the guess.

Outline

- Data Encryption Standard (DES)
  - Background
  - Description of DES
  - Security of DES
  - Differential and Linear Cryptanalysis
  - The Real Design Criteria
  - DES Variants
  - How Secure Is DES Today?

S-boxes Design Criteria

- After differential cryptanalysis became public, IBM published the design criteria for the S-boxes and the P-box. The criteria for the S-boxes are:
  - Each S-box has 6 input bits and 4 output bits. (This was the largest size that could be accommodated in a single chip with 1974 technology.)
  - No output bit of an S-box should be too close to a linear function of the input bits.
  - If you fix the left-most and right-most bits of an S-box and vary the 4 middle bits, each possible 4-bit output is attained exactly once.
  - If two inputs to an S-box differ in exactly 1 bit, the outputs must differ in at least 2 bits.
  - If two inputs to an S-box differ in the 2 middle bits exactly, the outputs must differ in at least 2 bits.
  - If two inputs to an S-box differ in their first 2 bits and are identical in their last 2 bits, the two outputs must not be the same.
  - For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.
  - A criterion similar to the previous one, but for the case of three active S-boxes.

S-boxes Design Criteria

- The criteria for the P-box are:
  - The 4 output bits from each S-box in round i are distributed so that 2 of them affect the middle-bits of S-boxes at round$_i$ + 1 and the other 2 affect end bits.
  - The 4 output bits from each S-box affect six different S-boxes; no 2 affect the same S-box.
  - If the output bit from one S-box affects a middle bit of another S-box, then an output bit from that other S-box cannot affect a middle bit of the first S-box.

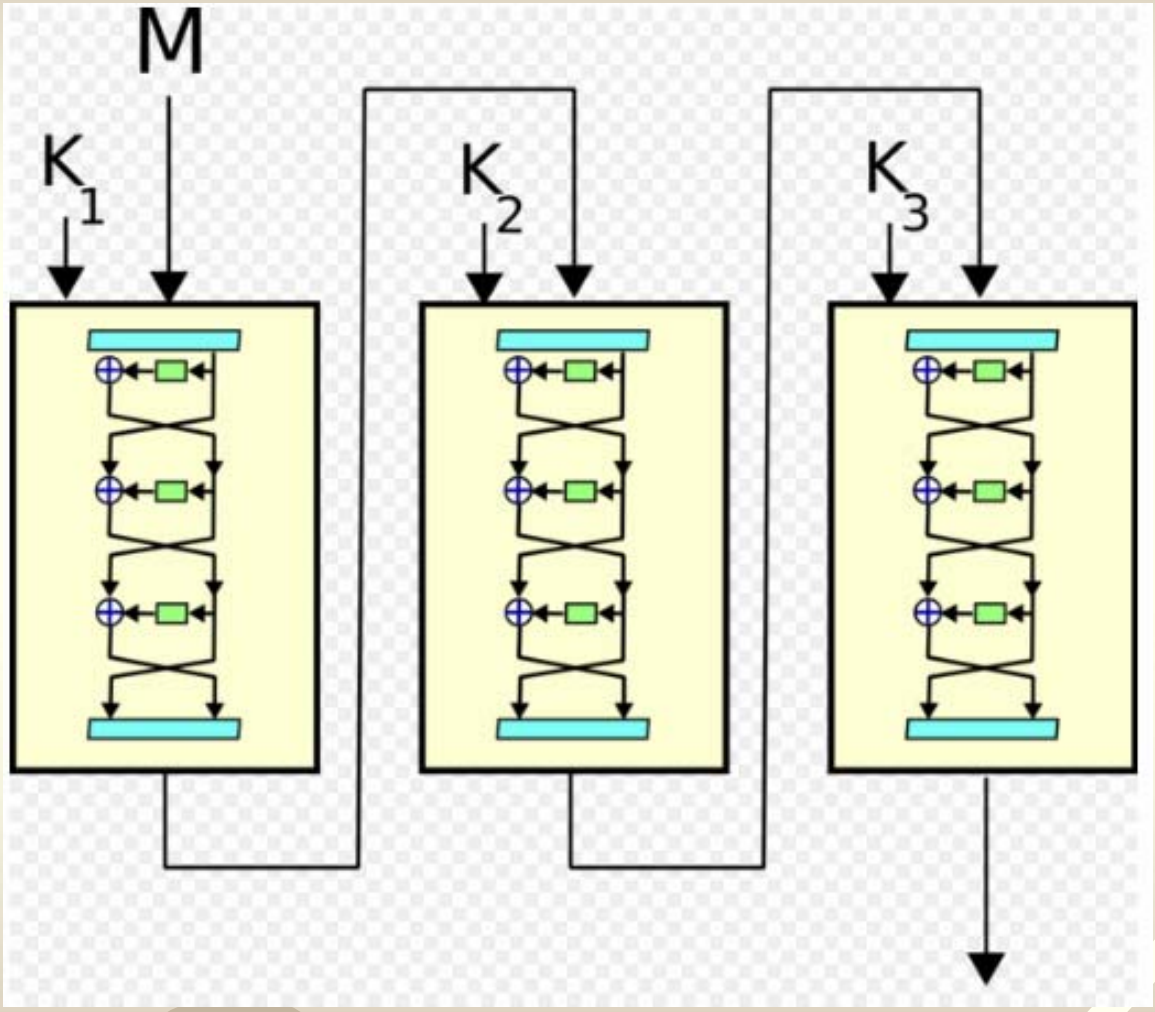
Outline

- Data Encryption Standard (DES)
  - Background
  - Description of DES
  - Security of DES
  - Differential and Linear Cryptanalysis
  - The Real Design Criteria
  - DES Variants
  - How Secure Is DES Today?

Multiple DES

- Some DES implementations use triple-DES.
- Since DES is not a group, then the resultant ciphertext is much harder to break using exhaustive search: 2^{112} attempts instead of 2^{56} attempts.

Triple DES



DES with Independent Subkeys

- Another variation is to use a different subkey for each round, instead of generating them from a single 56-bit key.
- Since 48 key bits are used in each of 16 rounds, this means that the key length for this variant is 768 bits.
- This variant would drastically increase the difficulty of a brute-force attack against the algorithm; that attack would have a complexity of 2^{768}.

Other DES Variants

- DESX is a DES variant from RSA Data Security, Inc. that has been included in the MailSafe electronic mail security program since 1986 and the BSAFE toolkit since 1987.
- CRYPT(3) is a DES variant found on UNIX systems. It is primarily used as a one-way function for passwords, but sometimes can also be used for encryption.
- Generalized DES (GDES) was designed both to speed up DES and to strengthen the algorithm. The overall block size increases while the amount of computation remains constant.
- And others …

Outline

- Data Encryption Standard (DES)
  - Background
  - Description of DES
  - Security of DES
  - Differential and Linear Cryptanalysis
  - The Real Design Criteria
  - DES Variants
  - How Secure Is DES Today?

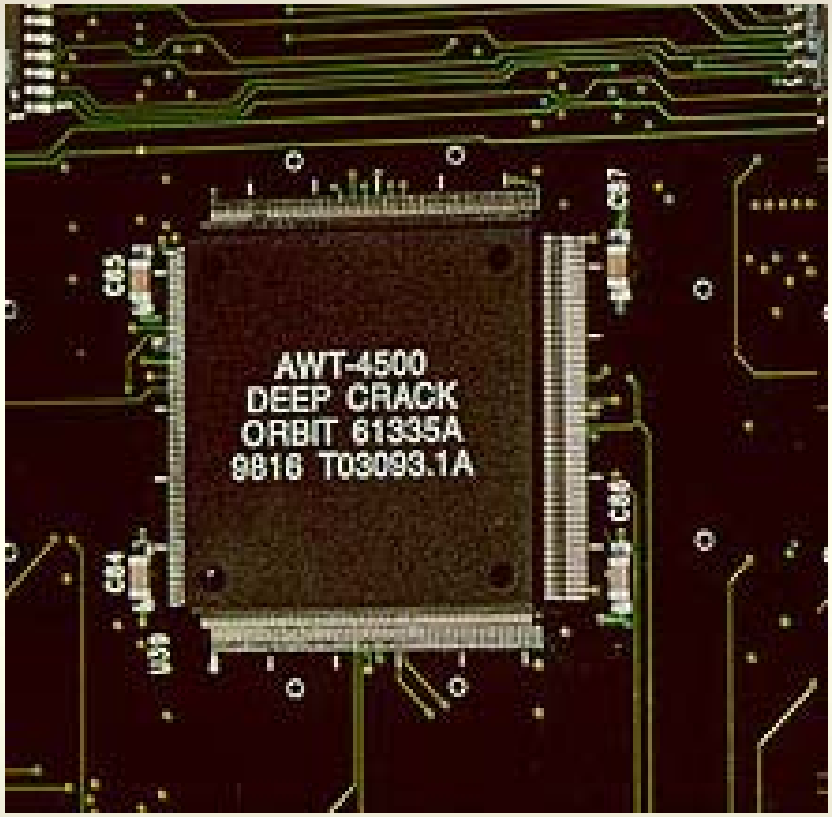
How Secure Is DES Today? 80-90s

- In academia, various proposals for a DES-cracking machine were advanced.
- In 1977, Diffie and Hellman proposed a machine costing an estimated US$20 million which could find a DES key in a single day.
- By 1993, Wiener had proposed a key-search machine costing US$1 million which would find a key within 7 hours.
- However, none of these early proposals were ever implemented — or, at least, no implementations were publicly acknowledged.
- The vulnerability of DES was practically demonstrated in the late 1990s.

How Secure Is DES Today? 90s

- In 1997, RSA Security sponsored a series of contests, offering a $10,000 prize to the first team that broke a message encrypted with DES for the contest. That contest was won by the DESCHALL Project, using idle cycles of thousands of computers across the Internet.
- The feasibility of cracking DES quickly was demonstrated in 1998 when a custom DES-cracker was built by the Electronic Frontier Foundation (EFF), a cyberspace civil rights group, at the cost of approximately US$250,000 (see EFF DES cracker).
- On July 17, 1998, Deep Crack decrypted a DES-encrypted message after only 56 hours of work, winning $10,000.
- This was the final blow to DES, against which there were already some published cryptanalytic attacks.

Deep Crack



- Advanced Wireless Technologies built 1856 custom ASIC DES chips (called *Deep Crack* or *AWT-4500*), housed on 29 circuit boards of 64 chips each.
- The boards were then fitted in six cabinets and mounted in a Sun-4/470 chasis.
- The search was coordinated by a single PC which assigned ranges of keys to the chips.
- The entire machine was capable of testing over 90 billion keys per second. It would take about 9 days to test every possible key at that rate. On average, the correct key would be found in half that time.

How Secure Is DES Today?

- In 2006, another custom hardware attack machine was designed based on FPGAs.
- COPACOBANA (COst-optimized PArallel COdeBreaker) shows a similar performance as Deep Crack at considerably lower cost.
  - This advantage is mainly due to progress in IC technology.

Advanced Encryption Standard

- The small key-space of DES, and relatively high computational costs of triple DES resulted in its replacement by AES as a Federal standard, effective May 26, 2002.
- **Advanced Encryption Standard (AES)** is an encryption standard adopted by the U.S. government.
- The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael**.**
- Each of these ciphers has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively.
- The AES ciphers have been analyzed extensively and are now used worldwide,

AES

- AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable (see Advanced Encryption Standard process for more details).
- The Rijndael cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and submitted by them to the AES selection process.
- It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce.
- It is available in many different encryption packages.
- AES is the first publicly accessible and open cipher approved by the NSA for top secret information.

AES Description

- AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.
- By 2006, the best known attacks were on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys.
- For cryptographers, a cryptographic "break" is anything faster than an exhaustive search.
- The largest successful publicly-known brute force attack has been against a 64-bit RC5 key by distributed.net.

AES: these days

- On July 1, 2009, Bruce Schneier blogged about a related-key attack on the 192-bit and 256-bit versions of AES, discovered by Alex Biryukov and Dmitry Khovratovich, which exploits AES's somewhat simple key schedule and has a complexity of $2^{99.5}$.
- In November 2009, the first known-key distinguishing attack against a reduced 8-round version of AES-128 was released as a preprint.

End of Lecture

- Readings
  - Book: Chapters 11 and 12