Security Engineering

Lesson 10

Combining Block Ciphers; Pseudo-Random-Sequence Generators and Stream Ciphers,

Spring 2010

Dr. Marenglen Biba

Combining Block Ciphers

Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

Combining block algorithms

- There are many ways to combine block algorithms to get new algorithms.
- The idea behind these schemes is to try to increase security without going through the trouble of designing a new algorithm.
- DES has been a secure algorithm However, the key is too short.
- Wouldn't it be nice to use DES as a building block for another algorithm with a longer key?
- We'd have the best of both worlds: the assurance of almost four decades of cryptanalysis plus a long key.

Multiple encryption

- Multiple encryption is one combination technique: using an algorithm to encrypt the same plaintext block multiple times with multiple keys.
- Cascading is like multiple encryption, but uses different algorithms.
- Encrypting a plaintext block twice with the same key, whether with the same algorithm or a different one, is not smart.
  - For the same algorithm, it does not affect the complexity of a brute-force search.
- If you are going to use any of the following techniques, make sure the multiple keys are different and independent.

Double Encryption

- A naïve way of improving the security of a block algorithm is to encrypt a block twice with two different keys.
- First encrypt a block with the first key, then encrypt the resulting ciphertext with the second key.
- Decryption is the reverse process.

$$C = E_{K2}(E_{K1}(P))$$

$$P = D_{K1}(D_{K2}(C))$$

Double Encryption

- The resultant doubly-encrypted ciphertext block should be much harder to break using an exhaustive search.
- Instead of 2^n attempts (where n is the bit length of the key), it would require 2^{2n} attempts.
- If the algorithm is a 64-bit algorithm, the doubly-encrypted ciphertext would require 2^{128} attempts to find the key.

Meet-in-the-middle attack

- This complexity turns out not to be true for a known-plaintext attack.
- Merkle and Hellman developed a time-memory trade-off that could break this double-encryption scheme in $2^n + 1$ encryptions, not in 2^{2n} encryptions.
- They showed this for DES, but the result can be generalized to any block algorithm.
- The attack is called a **meet-in-the-middle attack;**
  - ***It works by encrypting from one end, decrypting from the other, and matching the results in the middle.***

Meet-in-the-middle attack

- In this attack, the cryptanalyst knows P_1, C_1, P_2, and C_2, such that

$$C_1 = E_{K2}(E_{K1}(P_1))$$
$$C_2 = E_{K2}(E_{K1}(P_2))$$

- For each possible K, he computes $E_K(P_1)$ and stores the result in memory.
- After collecting them all, he computes $D_K(C_1)$ for each K and looks for the same result in memory.
- ***If he finds it, it is possible that the current key is K_2 and the key in memory is K_1.***
- He tries encrypting P_2 with K_1 and K_2; if he gets C_2 he can be pretty sure (with a probability of success of 1 in 2^{2m-2n}, where m is the block size) that he has both K_1 and K_2.

Complexity of the attack

- The maximum number of encryption trials he will probably have to run is $2*2^n$, or 2^{n+1}.
- If the probability of error is too large, he can use a third ciphertext block to get a probability of success of 1 in 2^{3m-2n}. There are still other optimizations.
- This attack requires a lot of memory: 2^n blocks. For a 56-bit algorithm, this translates to 2^{56} 64-bit blocks, or 10^{17} bytes.
- This is still considerably more memory storage than one could comfortably comprehend, but it's enough to convince the most paranoid of cryptographers that double encryption is not worth anything.

Davies-Price double encryption

- Another double-encryption method, sometimes called **Davies-Price**, is a variant of CBC.
  - $C_i = E_{K1}(P_i \oplus E_{K2}(C_{i-1}))$
  - $P_i = D_{K1}(C_i) \oplus E_{K2}(C_{i-1})$

- They claim “no special virtue of this mode,” but it seems to be vulnerable to the same **meet-in-the-middle attacks** as other double-encryption modes.

Combining Block Ciphers

Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

Triple Encryption with Two Keys

- A better idea, proposed by Tuchman, operates on a block three times with two keys: with the first key, then with the second key, and finally with the first key again.
- *He suggested that the sender first encrypt with the first key, then decrypt with the second key, and finally encrypt with the first key*.
- The receiver decrypts with the first key, then encrypts with the second key, and finally decrypts with the first key.
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - $P = D_{K1}(E_{K2}(D_{K1}(C)))$

Triple Encryption with Two Keys

- This is sometimes called **encrypt-decrypt-encrypt (EDE)** mode.
- If the block algorithm has an n-bit key, then this scheme has a $2n$-bit key. The curious encrypt-decrypt-encrypt pattern was designed by IBM.
- K_1 and K_2 alternate to prevent the meet-in-the-middle attack previously described.
  - If $C = E_{K2}(E_{K1}(E_{K1}(P)))$, then a cryptanalyst could precompute $E_{K1}(E_{K1}(P))$ for every possible K_1 and then proceed with the attack.
  - It only requires 2^{n+2} encryptions.

Triple encryption with three keys

- Triple encryption with two keys is not susceptible to the same meet-in-the-middle attack described earlier.
- But Merkle and Hellman developed another time-memory trade-off that could break this technique in 2^{n-1} steps using 2^n blocks of memory.
- If you are going to use triple encryption, Schneier recommends three different keys.
- The key length is longer, but key storage is usually not a problem. Bits are cheap.
  - $C = E_{K3}(D_{K2}(E_{K1}(P)))$
  - $P = D_{K1}(E_{K2}(D_{K3}(C)))$

Triple Encryption with Minimum Key (TEMK)

- The trick is to derive three keys from two: X_1 and X_2:
  - $K_1 = E_{X1}(D_{X2}(E_{X1}(T_1)))$
  - $K_2 = E_{X1}(D_{X2}(E_{X1}(T_2)))$
  - $K_3 = E_{X1}(D_{X2}(E_{X1}(T_3)))$

- $T1$, $T2$, and $T3$ are constants, which do not have to be secret.
- This is a special construction that guarantees that for any particular pair of keys, the best attack is a known-plaintext attack.

Combining Block Ciphers

Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

Doubling the Block Length

- There is some argument in the academic community whether a 64-bit block is long enough.
  - On the one hand, a 64-bit block length only diffuses plaintext over 8 bytes of ciphertext.
  - On the other hand, a longer block length makes it harder to hide patterns securely; there is more room to make mistakes.
- Some propose doubling the block length of an algorithm using multiple encryptions.
- Before implementing any of these, we have to look for the possibility of meet-in-the-middle attacks.
- However, Schneier advises against this sort of thing. It isn't faster than conventional triple encryption:
  - six encryptions are still required to encrypt two blocks of data.

Combining Block Ciphers

Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

Double OFB/Counter

- This method uses a block algorithm to generate two keystreams, which are then used to encrypt the plaintext.

$$S_i = E_{K1}(S_i - 1 \oplus I_1);\ I_1 = I_1 + 1$$

$$T_i = E_{K2}(T_i - 1 \oplus I_2);\ I_2 = I_2 + 1$$

$$C_i = P_i \oplus S_i \oplus T_i$$

- S_i and T_i are internal variables, and I_1 and I_2 are counters.
- The two keys, $K1$ and $K2$, are independent.

Quintuple Encryption

- If triple encryption isn't secure enough—perhaps you need to encrypt triple-encryption keys using an even stronger algorithm — then higher multiples might be in order.
- Quintuple encryption is very strong against meet-in-the-middle attacks.

$$C = E_{K1}(D_{K2}(E_{K3}(D_{K2}(E_{K1}(P)))))$$

$$P = D_{K1}(E_{K2}(D_{K3}(E_{K2}(D_{K1}(C)))))$$

Combining Block Ciphers

Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

CDMF Key Shortening

- This method was designed by IBM for their Commercial Data Masking Facility or CDMF to shrink a 56-bit DES key to a 40-bit key suitable for export.
- It assumes that the original DES key includes the parity bits.
  - **(1)** Zero the parity bits: bits 8, 16, 24, 32, 40, 48, 56, 64.
  - **(2)** Encrypt the output of step (1) with DES and the key 0xc408b0540ba1e0ae, and XOR the result with the output of step (1).
  - **(3)** Take the output of step (2) and zero the following bits: 1, 2, 3, 4, 8, 16, 17, 18, 19, 20, 24, 32, 33, 34, 35, 36, 40, 48, 49, 50, 51, 52, 56, 64.
  - **(4)** Encrypt the output of step (3) with DES and the following key: 0xef2c041ce6382fe6.
  - **This key is then used for message encryption.**
- Remember that this method shortens the key length, and thereby weakens the algorithm.

Combining Block Ciphers

Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

Whitening

- **Whitening** is the name given to the technique of XORing some key material with the input to a block algorithm, and XORing some other key material with the output.
- The idea is to prevent a cryptanalyst from obtaining a plaintext/ciphertext pair for the underlying algorithm.
- Since there is an XOR both before and after the block algorithm, this technique is not susceptible to a meet-in-the-middle attack.
  - $C = K_3 \oplus E_{K2}(P \oplus K_1)$
  - $P = K_1 \oplus D_{K2}(C \oplus K_3)$

Combining Block Ciphers

Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

Cascading Multiple Block Algorithms

- What about encrypting a message once with Algorithm A and key K_A, then again with Algorithm B and key K_B?
- Maybe Alice and Bob have different ideas about which algorithms are secure: Alice wants to use Algorithm A and Bob wants to use Algorithm B.
- This technique is sometimes called cascading, and can be extended far beyond only two algorithms and keys.

Cascading Multiple Block Algorithms

- If all of the multiple keys are **independent**, then the resultant cascade is at least as difficult to break as the first algorithm in the cascade.
- If the second algorithm is vulnerable to a chosen-plaintext attack, then **the first algorithm might facilitate that attack** and make the second algorithm vulnerable to a known-plaintext attack when used in a cascade.
- This potential attack is not limited to encryption algorithms: If you let someone else specify any algorithm which is used on your message before encryption, then you had better be sure that your encryption will withstand a chosen-plaintext attack.

Combining Block Ciphers

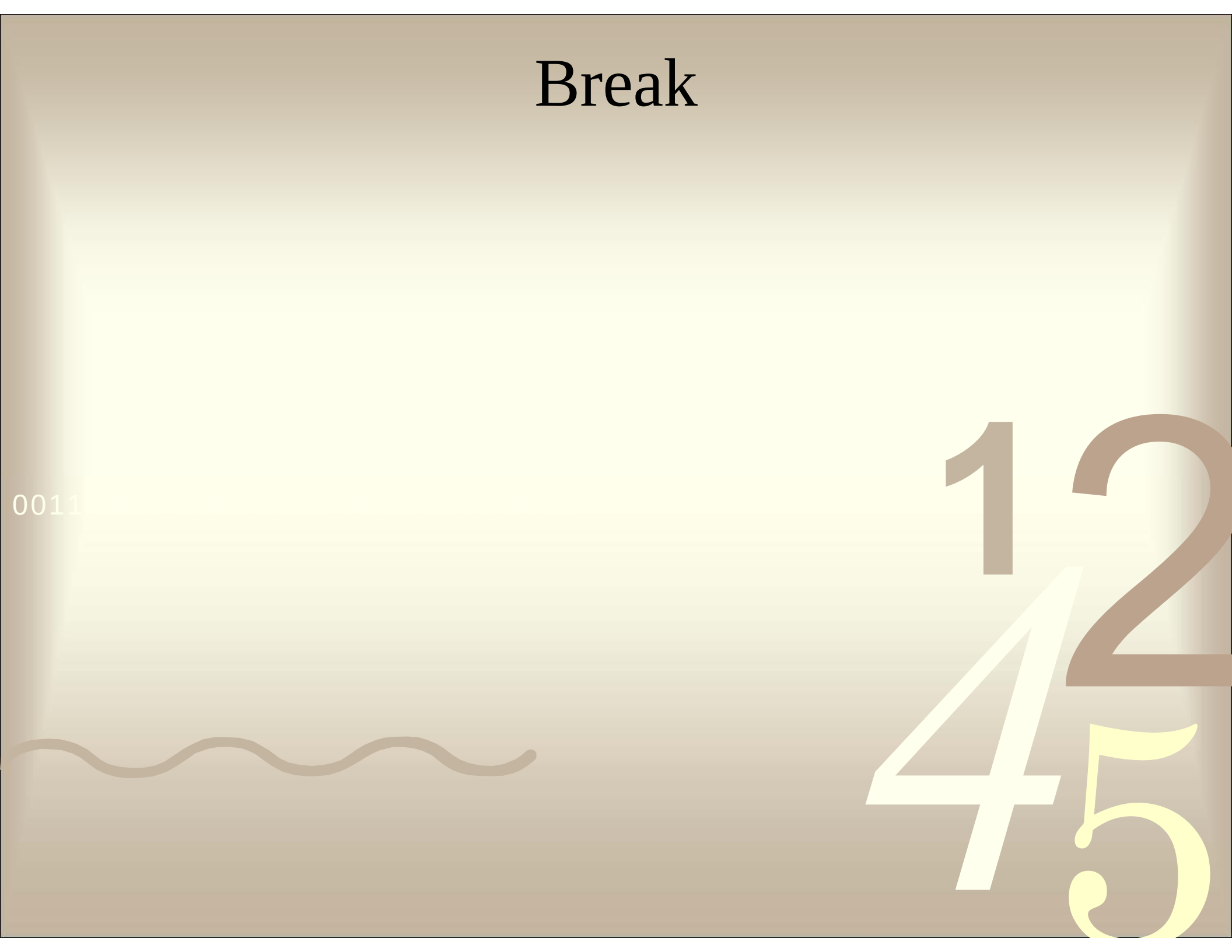
Combining Block Ciphers

- Double Encryption
- Triple Encryption
- Doubling the Block Length
- Other Multiple Encryption Schemes
- CDMF Key Shortening
- Whitening
- Cascading Multiple Block Algorithms
- Combining Multiple Block Algorithms

Combining Multiple Block Algorithms

- Here's another way to combine multiple block algorithms, one that is guaranteed to be at least as secure as both algorithms. With two algorithms (and two independent keys):
  - **(1)** Generate a **random-bit string**, *R*, the same size as the message *M*.
  - **(2)** Encrypt *R* with the first algorithm.
  - **(3)** Encrypt $M \oplus R$ with the second algorithm.
  - **(4)** The ciphertext message is the results of steps (2) and (3).
- Assuming the random-bit string is **indeed random**, this method encrypts *M* with a one-time pad and then encrypts both the pad and the encrypted message with each of the two algorithms.
- Since both are required to reconstruct *M*, a cryptanalyst must break both algorithms.

Break



Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Linear congruential generators

- **Linear congruential generators** are pseudo-random-sequence generators of the form
  - $X_n = (aX_n\text{-}1 + b) \bmod m$

  in which X_n is the nth number of the sequence, and X_{n-1} is the previous number of the sequence.
- The variables a, b, and m are constants: a is the multiplier, b is the increment, and m is the modulus. The key, or seed, is the value of X_0.
- This generator has a period **no greater than *m***.
- If a, b, and m are properly chosen, then the generator will be a **maximal period generator** and have period of m. (For example, b should be relatively prime to m.)

Breaking linear congruential generators

- Unfortunately, linear congruential generators cannot be used for cryptography; they are **predictable**.
- Linear congruential generators were first broken by Jim Reeds and then by Joan Boyar.
- She also broke quadratic generators:
  - $X_n = (aX_{n-1}^2 + bX_{n-1} + c) \bmod m$

  and cubic generators:
  - $X_n = (aX_n\text{-}13 + bX_{n-1}^2 + cX_{n-1} + d) \bmod m$
- Other researchers extended Boyar's work to **break any polynomial congruential generator**.
- Linear congruential generators remain useful for noncryptographic applications, however, such as **simulations**.

Combining Linear Congruential Generators

- Various people examined the combination of linear congruential generators.
- The results are no more cryptographically secure, but the combinations have longer periods and perform better in some randomness tests.

Outline

Chapter 16—Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Linear Feedback Shift Registers

- Shift register sequences are used in both cryptography and coding theory.
- There is a wealth of theory about them.
- Stream ciphers based on shift registers have been the workhorse of military cryptography since the beginnings of electronics.

Feedback Shift Registers

- A **feedback shift register** is made up of two parts: a **shift register** and a **feedback function**.
- The shift register is a sequence of bits. (The **length** of a shift register is figured in bits; if it is *n* bits long, it is called an *n*-bit shift register.)
- Each time a bit is needed, all of the bits in the shift register are shifted 1 bit to the right.
- The new left-most bit is computed as a function of the other bits in the register.
- The output of the shift register is 1 bit, often the least significant bit.

Feedback Shift Register

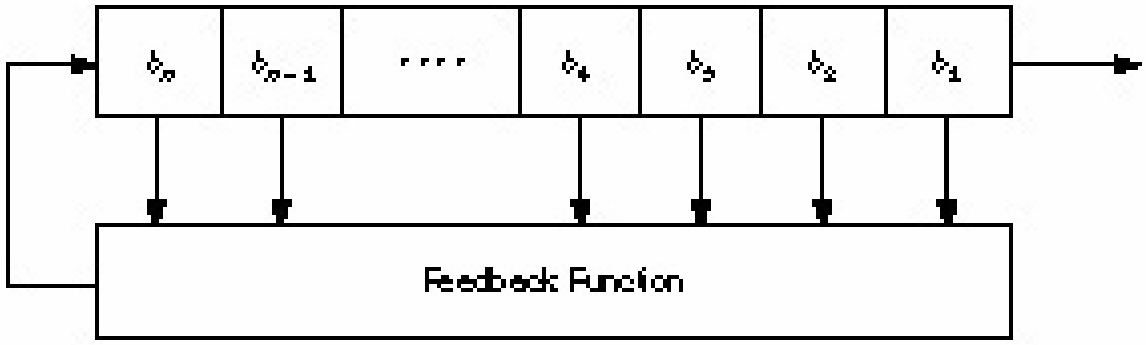


Figure 16.1 Feedback shift register.

- Cryptographers have liked stream ciphers made up of shift registers: They are easily implemented in digital hardware.

Linear feedback shift register

- The simplest kind of feedback shift register is a linear feedback shift register, or LFSR.
- The feedback function is simply the XOR of certain bits in the register;
  - the list of these bits is called a tap sequence.
- Sometimes this is called a Fibonacci configuration.
- Cryptographers like to analyze sequences to convince themselves that they are random enough to be secure.
- LFSRs are the most common type of shift registers used in cryptography.

Linear feedback shift register

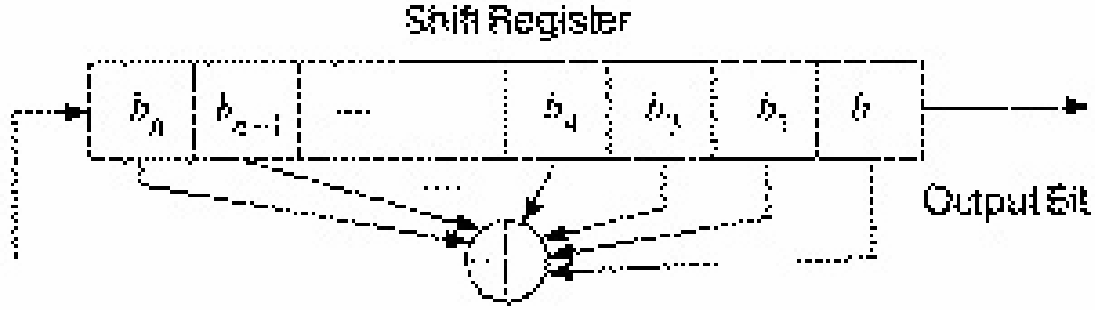


Figure 16.2 Linear feedback shift register.

4-bit LFSR

- If it is initialized with the value 1111, it produces the following sequence of internal states before repeating:
  - 1 1 1 1
  - 0 1 1 1
  - 1 0 1 1
  - 0 1 0 1
  - 1 0 1 0
  - 1 1 0 1
  - 0 1 1 0
  - 0 0 1 1
  - 1 0 0 1
  - 0 1 0 0
  - 0 0 1 0
  - 0 0 0 1
  - 1 0 0 0
  - 1 1 0 0
  - 1 1 1 0

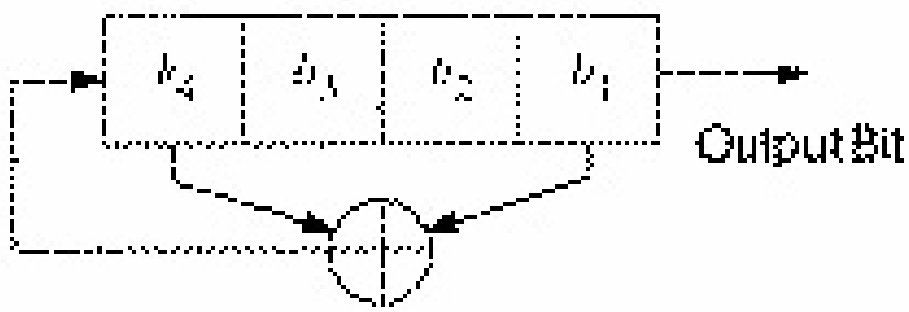


Figure 16.3 4-bit LFSR.

Randomness of LFSR

- An *n*-bit LFSR can be in one of $2^n - 1$ internal states.
- This means that it can, in theory, generate a $2^n - 1$-bit-long pseudo-random sequence before repeating.
- (It's $2^n - 1$ and not 2^n because a shift register filled with zeros will cause the LFSR to output a never-ending stream of zeros—this is not particularly useful.)

LFSRs in Software

- LFSRs are slow in software, but they're faster in assembly language than in C.

- One solution is to run 16 LFSRs (or 32, depending on your computer's word size) in **parallel**.

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Stream-ciphers and LFSRs

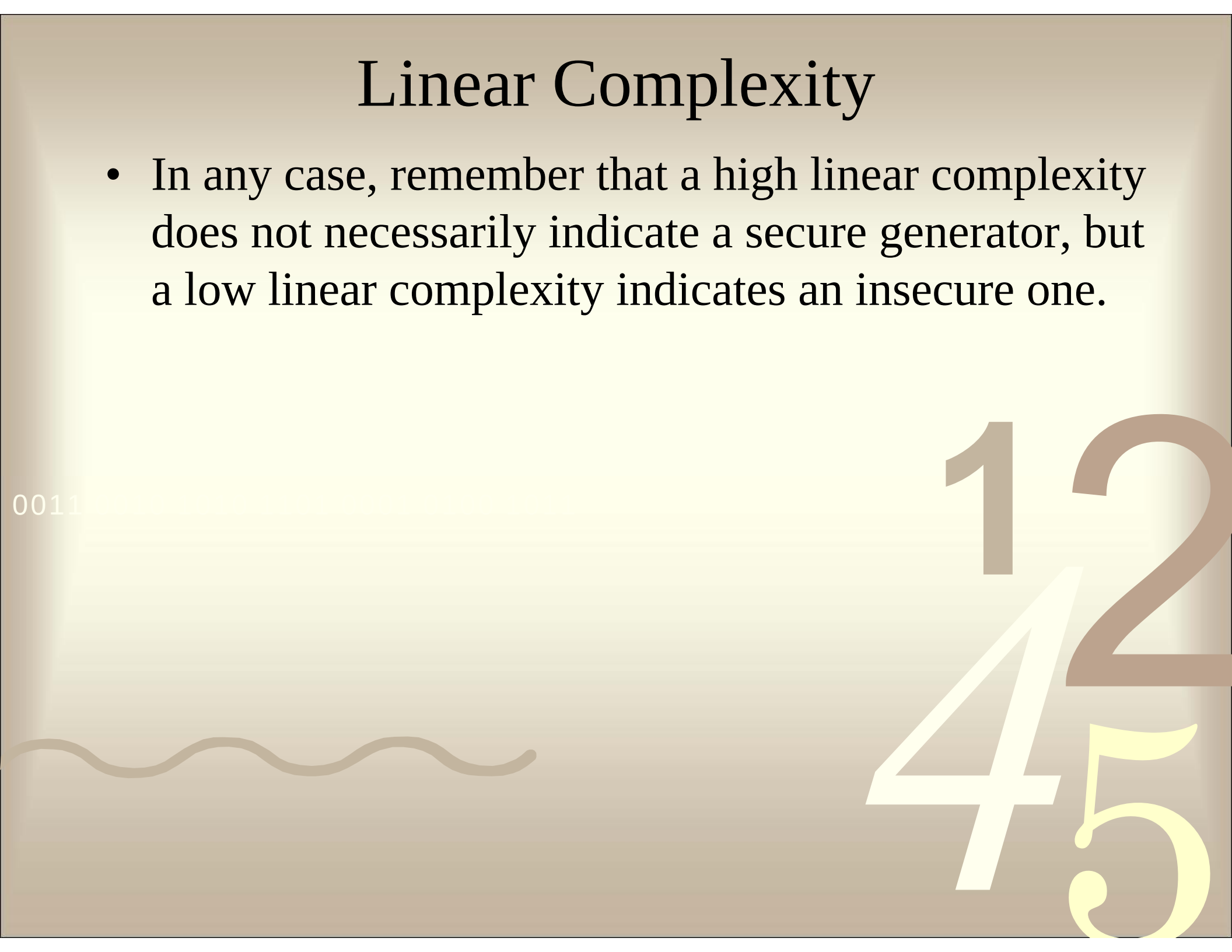
- Most practical stream-cipher designs center around LFSRs.
- In the early days of electronics, they were very easy to build.
- A shift register is nothing more than an array of bit memories and the feedback sequence is just a series of XOR gates.
- Even in VLSI circuitry, a LFSR-based stream cipher can give you a lot of security with only a few logic gates.
- The problem with LFSRs is that they are very inefficient in software.
- Any stream cipher outputs a bit at a time
  - you have to iterate the algorithm 64 times to encrypt what a single iteration of DES can encrypt.

Linear Complexity

- Analyzing stream ciphers is often easier than analyzing block ciphers.
- For example, one important metric used to analyze LFSR-based generators is linear complexity**.**
- This is defined as the length, *n*, of the shortest LFSR that can mimic the generator output.
- *Any sequence generated by a finite-state machine over a finite field has a finite linear complexity.*
- Linear complexity is important because a simple algorithm, called the Berlekamp-Massey algorithm, can generate this LFSR after examining only 2*n* bits of the keystream.
- Once you've generated this LFSR, you've broken the stream cipher.

Linear Complexity

- In any case, remember that a high linear complexity does not necessarily indicate a secure generator, but a low linear complexity indicates an insecure one.



Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Stream Ciphers Using LFSRs

- The basic approach to designing a keystream generator using LFSRs is simple.
- First you take one or more LFSRs, generally of different lengths and with different feedback polynomials. (If the lengths are all relatively prime and the feedback polynomials are all primitive, the whole generator is **maximal period**.)
- Every time you want a bit, shift the LFSRs once (this is sometimes called **clocking**).
- The output bit is a function, preferably a nonlinear function, of some of the bits of the LFSRs. This function is called the **combining function**, and the whole generator is called a **combination generator**.
- If the output bit is a function of a **single LFSR**, the generator is called a **filter generator**.

Geffe Generator

- This keystream generator uses three LFSRs, combined in a nonlinear manner.
- Two of the LFSRs are inputs into a multiplexer, and the third LFSR controls the output of the multiplexer.
- If a_1, a_2, and a_3 are the outputs of the three LFSRs, the output of the Geffe generator can be described by:

$$b = (a_1 \wedge a_2) \oplus ((\neg\, a_1) \wedge a_3)$$

- If the LFSRs have lengths n_1, n_2, and n_3, respectively, then the linear complexity of the generator is

$$(n_1 + 1)n_2 + n_1 n_3$$

- Although this generator looks good on paper, it is cryptographically weak and falls to a correlation attack

Generalized Geffe Generator

- Instead of choosing between two LFSRs, this scheme chooses between k LFSRs, as long as k is a power of 2. There are $k + 1$ LFSRs total.

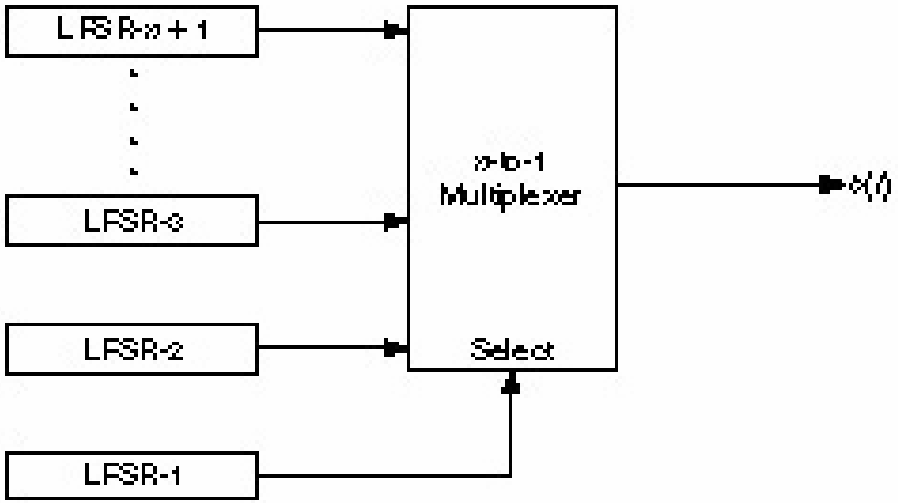


Figure 16.7 Generalized Geffe generator.

Generators

- Jennings Generator
- Beth-Piper Stop-and-Go Generator
- Alternating Stop-and-Go Generator
- Bilateral Stop-and-Go Generator
- Threshold Generator
- Multispeed Inner-Product Generator
- And many others…

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

A5

- A5 has been the stream cipher used to encrypt GSM (Group Special Mobile) a standard for digital cellular mobile telephones.
- It has been used to encrypt the link from the telephone to the base station.
- The rest of the link is unencrypted; the telephone company can easily eavesdrop on your conversations.

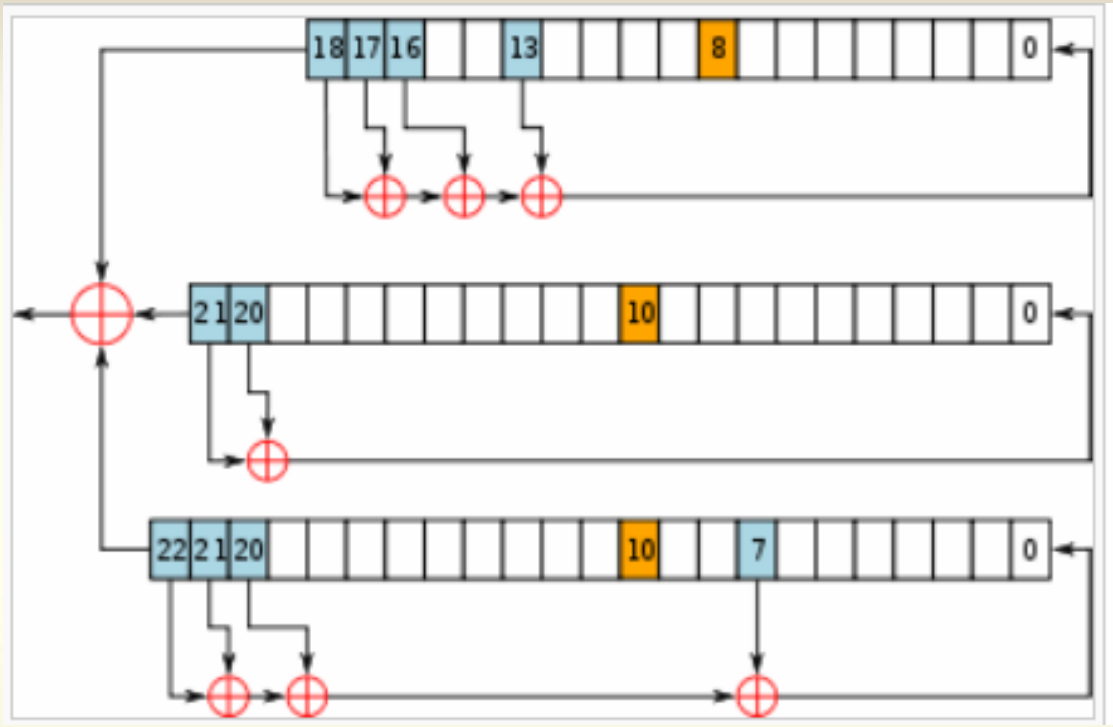
A5

- Originally it was thought that GSM's cryptography would prohibit export of the phones to some countries.
- Now some officials are discussing whether A5 might harm export sales, implying that it is so weak as to be an embarrassment.
- Ross Anderson reported in 1994 that the various NATO intelligence agencies had a catfight in the mid-1980s over whether GSM encryption should be strong or weak. The Germans wanted strong cryptography, as they were sitting near the Soviet Union. The other countries overruled them, and A5 is a French design.

A5

- A5 consists of three LFSRs; the register lengths are 19, 22, and 23.
- The output is the XOR of the three LFSRs.
- There is a trivial attack requiring 2^{40} encryptions: Guess the contents of the first two LFSRs, then try to determine the third LFSR from the keystream.
- It has become clear that the basic ideas behind A5 are good.
  - **It is very efficient.** It passes all known statistical tests;
  - Its only known weakness is that its registers are short enough to make exhaustive search feasible.
  - Variants of A5 with **longer shift registers** and denser feedback polynomials should be considered more secure.

A5/1



GSM and A5/1

- A GSM transmission is organised as sequences of *bursts*.
- In a typical channel and in one direction, one burst is sent every 4.615 milliseconds and contains 114 bits available for information.
- A5/1 is used to produce for each burst a 114 bit sequence of keystream which is XORed with the 114 bits prior to modulation.

A5/1 Today

- A similar effort, the A5/1 Cracking Project, was announced at the 2009 Black Hat security conference by cryptographers Karsten Nohl and Sascha Krißler.
  - It created the look-up tables using Nvidia GPGPUs via a peer-to-peer distributed computing architecture.
  - Starting in the middle of September 2009, the project ran the equivalent of 12 Nvidia GeForce GTX 260. According to the authors, the approach can be used on any cipher with key size up to 64-bits.

A5/1 Today

- In December 2009, the A5/1 Cracking Project attack tables for A5/1 were announced by Chris Paget and Karsten Nohl.
  - The tables use a combination of compression techniques, including rainbow tables and distinguished point chains.
  - These tables constituted only parts of the 2TB completed table, and had been computed during three months using 40 distributed CUDA nodes, and then published over BitTorrent.
  - More recently the project has announced a switch to faster ATI Evergreen (GPU family) code, and Frank A. Stevenson announced breaks of A5/1 using the ATI generated tables.

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Hughes XPD/KPD

- This algorithm is designed by Hughes Aircraft Corp.
- They put it in army tactical radios and direction-finding equipment for sale to foreign militaries.
- It was designed in 1986 and called XPD, for Exportable Protection Device. Later it was renamed KPD—Kinetic Protection Device—and declassified.
- The algorithm uses a 61-bit LFSR. There are 2^{10} different primitive feedback polynomials, which were approved by the NSA. The key selects one of these polynomials (they are all stored in ROM somewhere), as well as the initial state of the LFSR.
- This algorithm looks pretty impressive and the NSA allows export, so there must be some attack on the order of 2^{40} or less.

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Nanoteq

- Nanoteq is a South African electronics company.
- This is their algorithm that has been fielded by the South African police to encrypt their fax transmissions, and presumably for other uses as well.
- It uses a 127-bit LFSR with a fixed feedback polynomial; the key is the initial state of the feedback register.

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Rambutan

- Rambutan is a British algorithm, designed by the Communications Electronics Security Group (one of the aliases used by GCHQ).
- It is only sold as a hardware module and is approved for the protection of classified material up to “Confidential.”
- The algorithm itself is secret, and the chip is not generally commercially available.
- Rambutan has a 112-bit key (plus parity bits) and can operate in three modes: ECB, CBC, and 8-bit CFB.
- This strongly indicates that it is a block algorithm, but rumors point elsewhere.
- **Supposedly, it is a LFSR stream cipher.** It has five shift registers, each one of a different length around 80 bits.

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Additive generators

- **Additive generators** (sometimes called lagged Fibonacci generators) are extremely efficient because they produce **random words instead of random bits**.
- The initial state of the generator is an array of *n*-bit words: 8-bit words, 16-bit words, 32-bit words, whatever: X_1, X_2, X_3,..., X_m.
- This initial state is the key. The *i*th word of the generator is
  - $X_i = (X_{i-a} + X_{i-b} + X_{i-c} + ... + X_{i-m}) \bmod 2n$
- If the coefficients *a*, *b*, *c*,..., *m* are chosen right, the period of this generator is at least $2^n - 1$.
- One of the requirements on the coefficients is that the least significant bit forms a maximal-length LFSR.

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Gifford

- David Gifford invented a stream cipher and used it to encrypt news wire reports in the Boston area from 1984 until 1988.
- The algorithm has a single 8-byte register: b_0, b_1,..., b_7.
- The algorithm works in OFB.
- To generate a key byte k_i, concatenate b_0 and b_2 and concatenate b_4 and b_7.
- Multiply the two together to get a 32-bit number.
- The third byte from the left is k_i.

Gifford

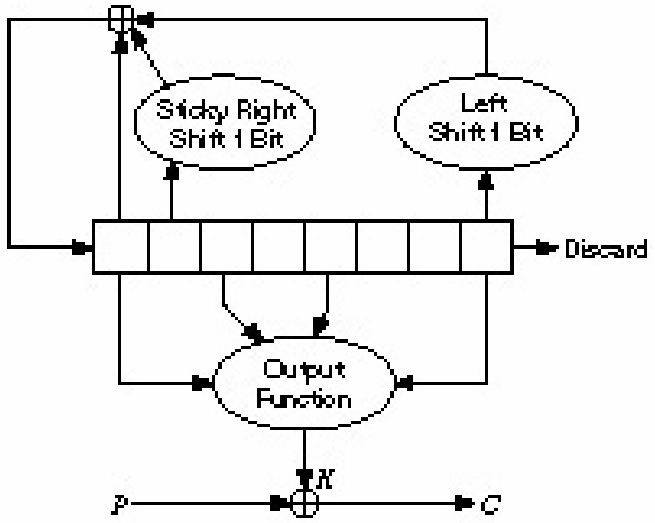


Figure 16.17 Gifford.

Outline

Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

Algorithm M

- The name is from Knuth.
- It’s a method for **combining multiple pseudo-random streams** that increases their security.
- One generator’s output is used to select a delayed output from the other generator.

Outline

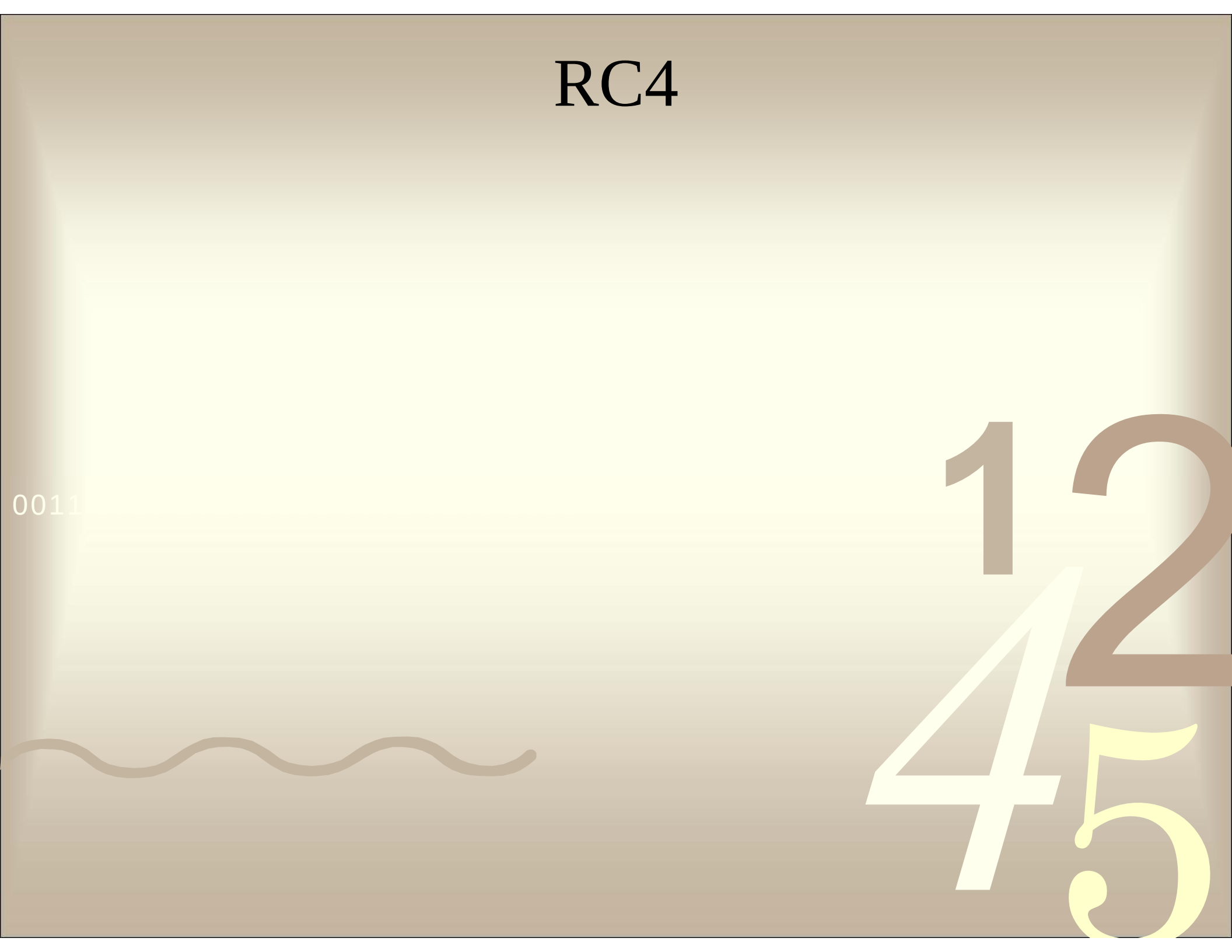
Pseudo-Random-Sequence Generators and Stream Ciphers

- Linear Congruential Generators
- Linear Feedback Shift Registers
- Design and Analysis of Stream Ciphers
- Stream Ciphers Using LFSRs
- A5
- Hughes XPD/KPD
- Nanoteq
- Rambutan
- Additive Generators
- Gifford
- Algorithm M
- PKZIP

PKZIP

- Roger Schlafly designed the encryption algorithm built into the PKZIP data compression program.
- It's a stream cipher that encrypts data one byte at a time.
- Unfortunately, it's not that great. An attack requires 40 to 200 bytes of known plaintext and has a time complexity of about 2^{27}.
  - You can do it in a few hours on your personal computer.
- If the compressed file has any standard headers, getting the known plaintext is no problem.

RC4



RC4

- In cryptography, **RC4** (also known as **ARC4** or **ARCFOUR** meaning Alleged RC4) is the most widely-used software stream cipher and is used in popular protocols such as Secure Sockets Layer (SSL) to protect Internet traffic and WEP ("wired equivalent privacy") to secure wireless networks.
- While remarkable for its simplicity and speed in software, **RC4 has weaknesses** that argue against its use in new systems.
- It is especially vulnerable when:
  - the beginning of the output keystream is not discarded
  - nonrandom or related keys are used
  - single keystream is used twice
- Some ways of using RC4 can lead to very insecure cryptosystems such as WEP.

RC4-based cryptosystems

- WEP
- WPA (default algorithm, but can be configured to use AES-CCMP instead of RC4)
- BitTorrent protocol encryption
- Microsoft Point-to-Point Encryption
- Secure Sockets Layer (optionally)
- Secure shell (optionally)
- Remote Desktop Protocol
- Kerberos (optionally)
- SASL Mechanism Digest-MD5 (optionally)
- PDF
- Microsoft Office
- Microsoft XBOX

RC4

- RC4 generates a pseudorandom stream of bits (a keystream) which, for encryption, is combined with the plaintext using bit-wise XOR; decryption is performed the same way (since XOR is a symmetric operation).
- To generate the keystream, the cipher makes use of a secret internal state which consists of two parts:
  - A permutation of all 256 possible bytes.
  - Two 8-bit index-pointers (denoted "i" and "j").
- The permutation is initialized with a variable length key, typically between 40 and 256 bits, using the *key-scheduling* algorithm (KSA).
- Once this has been completed, the stream of bits is generated using the *pseudo-random generation algorithm* (PRGA).

RC4 Attacks: Fluhrer, Mantin and Shamir attack

- In 2001, a new and surprising discovery was made by Fluhrer, Mantin and Shamir:
  - Over all possible RC4 keys, the statistics for the first few bytes of output keystream are strongly non-random, leaking information about the key.
- This and related effects were then used to break the WEP encryption used with 802.11 wireless networks. This caused a scramble for a standards-based replacement for WEP in the 802.11 market, and led to the **IEEE 802.11i** effort and WPA.
- Cryptosystems can defend against this attack by discarding the initial portion of the keystream.
- Such a modified algorithm is traditionally called "RC4-drop[n]", where n is the number of initial keystream bytes that are dropped. The default is n = 768 bytes, but a conservative value would be n = 3072 bytes

Klein's Attack

- In 2005, Andreas Klein presented an analysis of the RC4 stream cipher showing more correlations between the RC4 keystream and the key.
- Erik Tews, Ralf-Philipp Weinmann, and Andrei Pychkine used this analysis to create aircrack-ptw, a tool which cracks 104-bit RC4 used in 128-bit WEP in under a minute.
- Whereas the Fluhrer, Mantin, and Shamir attack used around 10 million messages, aircrack-ptw can break 104-bit keys with less.

End of Lecture

- Readings
  - Applied Crypto: Chapters 15, 16.