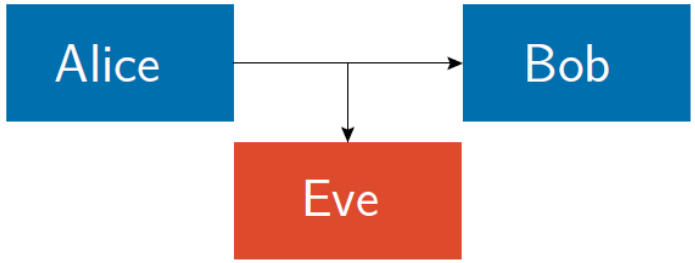
Week 2

Ciphers

THE UNIVERSITY OF SYDNEY

Cryptography

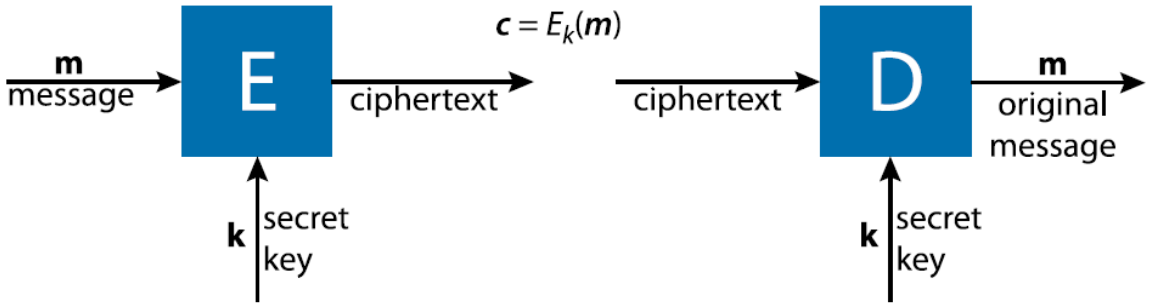
- **Cryptography** is the study of mathematical techniques related to the design of ciphers.
- The fundamental application of cryptography is enabling secure communications over an insecure channel.
- How can Alice send a secure message to Bob over an insecure channel when Eve is listening in? Eve is an active attacker and may tap, insert or modify messages in transit.
- How does one use cryptography to provide security such as:
  - Authentication
  - Confidentiality
  - Integrity
  - Non-Repudiation



Symmetric Ciphers

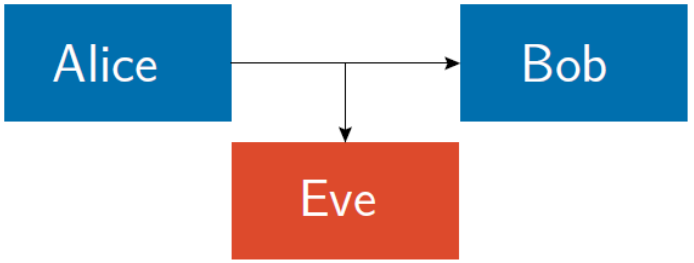
- The traditional way of achieving secrecy is through a shared secret key. This is also known as symmetric encryption since the key used to both encrypt and decrypt messages is the same.
- A symmetric cipher is an encryption algorithm E_k and a decryption algorithm D_k which inverts E_k. In other words, for all keys k and messages m

$$D_k(E_k(m)) = m$$



Symmetric Ciphers

- Alice and Bob share:
  - A secret key: k
  - An encryption algorithm: E_k
  - A decryption algorithm: D_k



- Alice wants to send Bob the message m. The unencrypted message is known as either the plaintext or cleartext.

- Alice encrypts m by computing the ciphertext c:

$$c = E_K(m)$$

- Bob decrypts c by computing the original plaintext message m:

$$D_k(c) = m$$

- A cryptosystem is a system consisting of an algorithm, plus all possible plaintexts, ciphertexts, and keys.

Symmetric Ciphers

- It is computationally hard to decrypt the ciphertext c without the secret key k. The secret key k is usually a large number of bits ($\geqslant 128$). The range of possible values of k is called the **key space** K. The range of possible messages is called the **message space** M.

- For 128-bit keys, the key space is $K = \{0,1\}^{128}$, i.e., $\{0,1,\ldots,2^{128}-1\}$.
  - Fun fact: The earth is about 4.5 billion years old. If we had been brute forcing keys at a rate of 1 billion keys per second since the earth was formed, we would have by now assessed less than one trillionth of the key space.

- Two main types of symmetric ciphers:
  - Stream Ciphers
    - Operate on a single bit or byte at a time.
  - Block Ciphers
    - Operate on blocks (numbers of bits) of plaintext at a time.

Cryptanalysis

- Cryptanalysis is used to breach cryptographic security systems and gain access to the contents of encrypted messages, even if the cryptographic key is unknown.

- We always assume that attackers have:
  - Complete access to the communications channel
  - Complete knowledge about the cryptosystem

- Secrecy must only depend on the key.

Cryptanalysis Attacks: COA

- Ciphertext Only Attack (COA)
  - Attacker only has access to the ciphertext
  - Given

$$c_1 = E_k(m_1), c_2 = E_k(m_2), \ldots, c_n = E_k(m_n)$$

  - Find any of
    - Any message $m_1, \ m_2, \ \ldots, \ m_n$
    - Secret key k
    - An algorithm to infer m_j from c_j

Cryptanalysis Attacks: KPA

- Known Plaintext Attack (KPA)
  - Attacker intercepts a random plaintext / ciphertext pair: (m, c)
  - Given

    $[m_1, c_1 = E_k(m_1)]$, $[m_2, c_2 = E_k(m_2)]$, …, $[m_n, c_n = E_k(m_n)]$

  - Find any of
    - Secret key k
    - An algorithm to infer m_j from c_j
  - Examples:
    - An attacker knowing that source code is being encrypted, the first bytes are likely #include, copyright notices, etc.

Cryptanalysis Attacks: CPA

- Chosen Plaintext Attack (CPA)
  - Attacker selects a message m and receives a ciphertext c
  - Stronger than KPA – Some ciphers resistant to KPA are not resistant to CPA
  - Given:

$$[m_1, c_1 = E_k(m_1)], \ldots, [m_n, c_n = E_k(m_n)] \text{ with chosen } m$$

  - Find any of
    - Secret key k
    - An algorithm to infer m_j from c_j
  - Example:
    - Feed specific intelligence to an adversary with the goal that it ends up encrypted, sent back and intercepted.

Cryptanalysis Attacks: CCA

- Chosen Ciphertext Attack (CCA)
  - Attacker specifies a ciphertext c and receives the plaintext message m
  - Given:

$$[c_1, m_1 = D_k(c_1)], \ldots, [c_n, m_n = D_k(c_n)] \text{ with chosen } c$$

  - Find
    - Secret key k

Classes of Break

From worst to least severe:

- **Total Break**: Attacker finds secret key k and hence can compute ***all*** decrypted messages $D_k(c)$ and perform encryption $E_k(m)$.

- **Global Deduction:** Attacker finds alternate algorithm A, equivalent to decrypting all messages $D_k(c)$ ***without*** finding k.

- **Local Deduction:** Attacker finds or decrypts the plaintext of ***one*** intercepted ciphertext

- **Information Deduction:** Attacker gains ***some*** information about the key or plaintext e.g. first few bits of a file, meaning of a message, file type, etc.

Attack Metrics

- An algorithm is **unconditionally secure** if no matter how much ciphertext an attacker has, there is not enough information to deduce the plaintext.
- Information security is a resource game with attacks measured in terms of:
  - Data Requirements
    - How much data is necessary to succeed?
  - Processing requirements (work factor)
    - How much time is needed to perform the attack?
  - Memory requirements
    - How much storage space is required?
  - Computational cost
    - How many GPU instances required?

Substitution Ciphers

- Substitution ciphers are the oldest form of cipher. The secret key consists of a table which maps letter substitutions between plaintext and ciphertext.
- The most famous is the Caesar cipher where each letter is shifted by 3 (modulo 26):

```
abcdefghijklmnopqrstuvwxyz
DEFGHIJKLMNOPQRSTUVWXYZABC
```

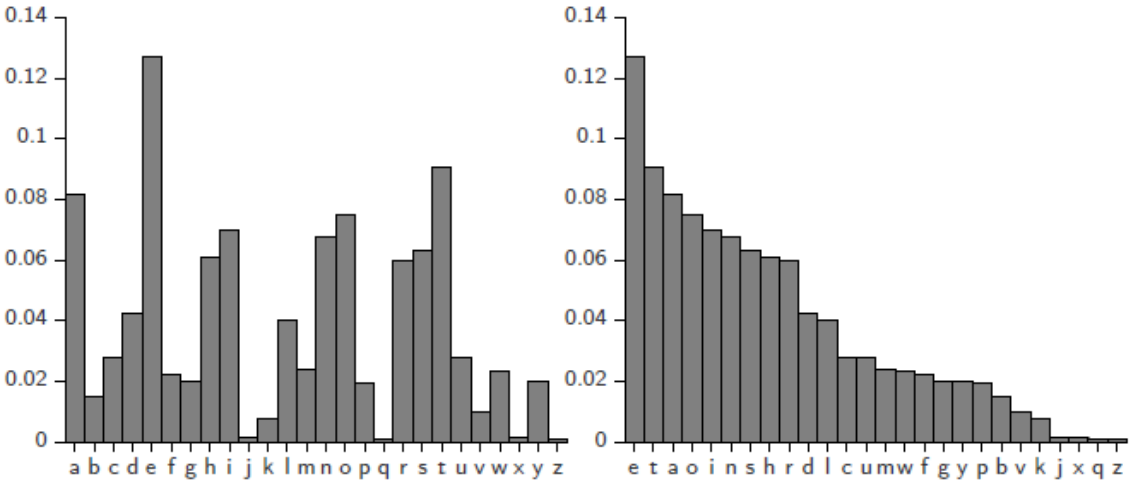
  - Similar to ROT13 which shifts plaintext 13 places – largest advantage is that encrypting twice results in the plaintext: $\mathrm{ROT13}(\mathrm{ROT13}(m)) = m$
- There are 26! (factorial) different possible keys ($\approx 2^{88}$ or 88-bits). Monoalphabetic (single character) substitution cipher:

```
Src = abcdefghijklmnopqrstuvwxyz
Key = XNYAHPOGZQWBTSFLRCVMUEKJDI

  m = thiscourserockstheblock
  c = MGZVYFUCVHCFYWVMGHNBFYW
```

Substitution Ciphers

- Substitution ciphers are easy to break (ciphertext only attack) using frequency analysis of the letters:
  - Single letters
  - Digraphs (pairs of letters)
  - Trigraphs (three letters)



The frequency of English letters.

Improved Substitution Ciphers: Homophonic Ciphers

- Homophonic ciphers are substitution ciphers that replace a common letter with multiple symbols (i.e. E can go to [C, ε, O])
- Peaks or troughs in the letter frequency are hidden as they are broken down into multiple smaller spikes.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
=================================================
D X S F Z E H C V I T P G A Q L K J R U O W M Y B N
9       7       3         5 0       4 6
        2
        1
```

  - "D" would become "F"
  - "E" would be randomly chosen as one of: [Z, 7, 2, 1].
  - The high frequency of the letter "E" (the most common letter in English) is spread amongst several characters, making frequency analysis much more difficult.

Permutation Ciphers

- Permutation ciphers are also known as transposition ciphers. The secret key, π, is a random permutation.
- Given a message $m = [m_1, m_2, m_3, \ldots, m_n]$
  - We can compute the encryption via: $E_\pi(m) = [m_{\pi(1)}, m_{\pi(2)}, m_{\pi(3)}, \ldots, m_{\pi(n)}]$
  - Suppose π is:

1	2	3	4	5	6
4	3	1	5	2	6

  - Then E_π("crypto") ="PYCTRO"

Vigenere Cipher

- Originated in Rome in the sixteenth century, a Vigenere cipher is a polyalphabetic substitution cipher (made of multiple monoalphabetic substitution ciphers).
- The secret key is a repeated word with encryption performed by adding the key modulo 26.

```
Plaintext:   launchmissilesatlosangeles
Keystream:   cryptocryptocryptocryptocr
=====================================
Ciphertext:  nrscvvozqhbzgjyiecurlvwzgj
```

  - L + C = 11 + 2 = 13 mod 26 = 13th char ⇒ N
  - N + Y = 13 + 24 = 37 mod 26 = 11th char ⇒ L
  - Note: The zero index (0th character) is A.
- Punctuation and white space are removed to increase cryptanalysis difficulty.

Breaking Vigenere Ciphers: Index of Coincidence

- Vigenere ciphers can be distinguished from simple substitution ciphers, using the index of coincidence.
- Index of coincidence is a statistical measure that assesses the probability of identical letters appear in the same position of two random texts.
- The index of coincidence κ is calculated by the following formula:

$$\kappa = \frac{1}{N(N-1)} \sum_{i=A}^{i=Z} F_i(F_i - 1)$$

  - F_i is the count of letter i (where i = A, B, . . . , Z) in the ciphertext, and N is the length of the ciphertext.

Index of Coincidence

- The index of coincidence can be used to detect the **length of the key** in a Vigenere cipher, by use of the following observation:
  - A substitution cipher does not change the index of coincidence. Since a substitution cipher only rearranges the terms in the sum, it does not change the index of coincidence.

- Using standard frequencies with which individual letters appear in English, the probability that a coincidence will occur is approximately $K_p = 0.0667$.

- If the text is random and letters chosen with equal probability, then the probability of a coincidence is much smaller: $K_p = 0.0385$ $(= 1/26)$.

Example: Breaking Vigenere

```
TPCTY LVEOO GBVRC BTWXS IHDKD QIRVQ QUKWL TMNQO EKMLP AURKL VHIUX YJRNV QWJEK UEQVD IXPLU RKLVT QSLKI LWAZI JWXPL
QRKIO PWFME XLLCP KDIKV EUXYX EAAQV MEKVN AVZRQ JGEMX LQUPM PCRLO IZPZZ FPONI AYPVQ RMVHC QZFLV IKGUK LRXER MDWVG
RVMPQ PWLWT TIYEQ JAYMK XBPUK PZJBF WIRKS QJMSV FYKFP QLRXE OIPID IQQLI ICPFP LMVBR BUAMW KLLUM FLRXE CDQFV IKNWZ
KUIXF BTIII CQZQM JQVUX UVZXL XMDAY IIOMP AZXEK VYIDC EGIDX NMQJQ ZQVMP FMESC EQGQD IDIJD MDXYI ACGES WSIFQ YIUMQ
CBQSE EINBT CNSOM AUQLW BQVFL VALTS AJKLV JIZHJ MPVZQ XTLCQ ZFLDC ECVPW LRQQB TIVQV UWGPK LFTAF IKLXH BQVKL BGIEE
KLFTA FCCEK FAQPR LEGID QVWMG MPMCC LNWDH DCPRQ DMKJX KTQXY LFFMZ SKXEA NMGVJ OQUYI CIPVQ NICMH GCZXF XEGUF LRXDQ
LAAEM KVWFL VTFVK MYJIJ GBALV EOVPK PFZFP OWMEH KGAEM EXEGU AVEMK INAVZ RQJMQ HFMQT CEXTE RUMYI KSHPW IXYIT CGILV
VBKVU WYSRN LIECO CQZUP ZJQWX YCJSR NCZXF XEGMP ICMSG ZYIFP LTLRV FQJKV QIEIJ KMEMW PBGCZ XFXEG MFSYM AGUQX VEZJU
QXFHL VPKAZ PIHWD XYSRC ZFQPK LFBTC JTFTQ FMJKL QLXIR HJGQZ XFXEG TMRUS CWXDM XLQPM EWHYF ESQRD ILNWD HWSOV PKRRQ
BUAMO VJLTB TCIMD JBQSL WKGAE WROBD ZURXQ VUWGP FYQQN FVFYY NMMRU SCVPK QVVZA KGXFJ COQZI VRBOQ QWRRA FMEXI SVCTX
XYIJV PMXRJ CNQOX DCPQC XJFVF CUFLP WBTDM RK
```

- Slice the ciphertext into N slices, by picking every N^{th} letter. Then, find the average index of coincidence across slices.
- Repeat for various choices of N.

1: .028	5: .042
2: .045	6: .035
3: .034	**7: .070**
4: .037	8: 0.32

- Key length is most likely 7, since it is closest to 6.67%

Example: Breaking Vigenere

- Once the key length N is known, we independently attack the N slices of the message. Each of the N slices of ciphertext C is an independent monoalphabetic substitution cipher:

$$(C_0, C_N, C_{2N}, \ldots),$$
$$(C_1, C_{N+1}, C_{2N+1}, \ldots),$$
$$\ldots$$
$$(C_{N-1}, C_{N+(N-1)}, C_{2N+(N-1)}, \ldots)$$

- Note the index of coincidence varies by language and can be domain specific (e.g., may be noticeably different for a physics journal paper).

Basic Cipher Operations: XOR and OTP

THE UNIVERSITY OF SYDNEY

What is XOR?

- XOR is the "exclusive or" operation: one or the other, but not both. It is addition modulo 2 and is represented by $\oplus$: $a \oplus b = (a + b) \, mod \, 2$

a	b	a&b	a||b	a $\oplus$ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Table: Truth Table with XOR

- Something XOR'd with itself is zero: $A \oplus A \equiv 0$
- XOR is associative: $A \oplus (B \oplus C) \equiv (A \oplus B) \oplus C$
- and commutative: $A \oplus B \equiv B \oplus A$

XOR Encryption

- XOR (addition modulo 2) is commonly used to provide security in programs. It is very weak by itself but forms the building block of most crypto primitives.

- Typically, we XOR bits together where the plaintext is XOR'd with a key stream to produce cipher text

```
Plaintext:   011011000110111101101100
Keystream:   011000110110000101110100
=================================
Ciphertext:  000011110000111000011000
```

- This is the same as a Vignere cipher (addition modulo 26) where XOR is addition modulo 2 since we are dealing with bits not letters

XOR Encryption

- The message m is XOR'd bitwise with a secret key:
  - $c = m \oplus k$
  - $m = c \oplus k$

- Since XOR is effectively a Vigenere cypher, it is easy to break:
  - Determine the key length N from index of coincidence
  - Shift cyphertext by N and XOR with itself

    This removes the key

    $$c \oplus c' = m \oplus k \oplus m' \oplus k = m \oplus m'$$
  - Results in message XOR'd with a shifted version of itself
  - Language is extremely redundant (English 1.3 bits of information/byte)
  - Easy to then decrypt

One Time Pad (OTP)

- A one-time pad is using a different substitution cipher for each letter of the plaintext.
- A one-time pad is perfectly secure provided that:
    - The secret key, k, is truly random
    - The plaintext does not repeat
    - The keystream does not repeat
- Failure to meet any one of these requirements results in zero security.
- The strength comes from the fact that a ***truly random key*** added to plaintext, produces a truly random ciphertext.
- No amount of computing power can break a one-time pad.
    - brute force would yield each and every possible message of that length.

One Time Pad (OTP)

- Core Problems: key distribution, key destruction, synchronisation.
  - k must be same length as m:
    - to encrypt 1GB you need a 1GB shared key.
  - Used for ultra-secure, low bandwidth communications
    - e.g., military satellites, Moscow-Washington phone line
  - Future: Quantum Key Distribution
    - secure distribution at a distance.

Perfect Secrecy

- Goal of cryptography:
  - **ciphertext reveals nothing about the plaintext.**

- A cipher has perfect secrecy if, for all $m \in M, c \in C$, the plaintext and ciphertext are statistically independent:

$$\Pr[\mathcal{M} = m | \mathcal{C} = c] = \Pr[\mathcal{M} = m]$$

- Few ciphers possess perfect secrecy
  - Reason: the key is much shorter than the message, in most ciphers.

Breaking OTP: Two Time Pad

- A two-time pad is perfectly insecure. Suppose two messages m_1, m_2 are encrypted using the same key k:
  - $c_1 = m_1 \oplus k$
  - $c_2 = m_2 \oplus k$
- Then the key k may be cancelled by XORing the ciphertexts:

$$\begin{aligned} c_1 \oplus c_2 &= (m_1 \oplus k) \oplus (m_2 \oplus k) \\ &= m_1 \oplus m_2 \oplus k \oplus k \\ &= m_1 \oplus m_2 \end{aligned}$$

- $m_1 \oplus m_2$ can be easy to separate due to redundancy in English and in ASCII (for example, bit 6 is set in letters but not most punctuation).

Breaking OTP: Malleability Attack

- The OTP and all stream ciphers are highly malleable. Suppose plaintext is a one-bit vote $v \in 0, 1$
  - $v = 0$ is a vote for Labor
  - $v = 1$ is a vote for Liberal
- Alice encrypts her vote using OTP and sends to Bob:
  - $c = v \oplus k$ where $k \in 0, 1$ is randomly chosen
- Mallory intercepts the ciphertext and sends with bits flipped:
  - $c' = c \oplus 1 = !c$
- Bob receives c′ and decrypts vote:

$$\begin{aligned} c' \oplus k &= c \oplus 1 \oplus k \\ &= v \oplus k \oplus 1 \oplus k \\ &= v \oplus 1 \\ &= !v \end{aligned}$$

PRNGs and Stream Ciphers

THE UNIVERSITY OF SYDNEY

Pseudorandom Number Generators (PRNGs)

- A source of random numbers is essential in many occasions:
  - Session Keys
  - Shuffling of Cards
  - Challenges
  - Nonces
- Computers are inherently **deterministic.** As a result, true randomness is a difficult thing to come by.
- Since we cannot get randomness easily, we use **p**seudo-**r**andom **n**umber **g**enerator functions (PRNGs) to generate what **appears** to be **statistically random** output.
- A **c**ryptographically **s**ecure **p**seudo-**r**andom **n**umber **g**enerator (CSPRNG) is a type of PRNG whose properties make it suitable for use in cryptography.

Sourcing Randomness

- PRNG functions produce the same sequence of seemingly random output when provided with a particular **"seed"** data.
- Since a computer is deterministic it must **extract randomness (entropy)** from an external, truly random source. This could be something like:
  - Thermal noise of hard drives
  - Low-order bit fluctuations of voltage readings
  - User input
  - Geiger counter click timing
- Randomness can actually be really hard to come by.
  - US5732138A - Method for seeding a pseudo-random number generator with a cryptographic hash of a digitization of a chaotic system - Google Patents

Properties of PRNGs

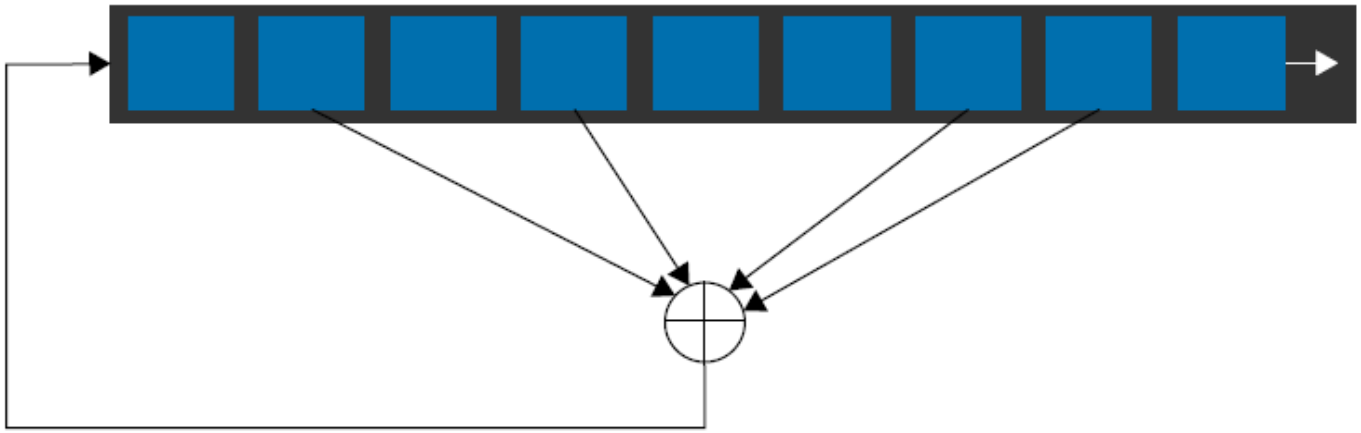
- Desirable properties of PRNGs include:
  - Repeatability
  - Statistical randomness
  - Long period/cycle
  - Computational efficiency
- PRNGs are often broken by:
  - Statistical tests that find patterns or biases in the output sequence
  - Inferring the state of the internal registers from the output sequence
- PRNGs are usually critically important parts of a cryptosystem. They are often a single point of failure.
  - Android Security Vulnerability (bitcoin.org)

Linear Congruential Generators (LCGs)

- An LCG generates a sequence $x_1, x_2, \ldots$ by starting with a seed x_0 and using the rule:
  - $x_{n+1} = (ax_n + b) \, mod \, c$
  - where a, b, and c are fixed constants.
    - The period of the PRNG is at most c.
    - Vulnerable to attacks since only three values x_i, x_{i+1}, x_{i+2} are needed to determine a and b.
- Advantages:
  - Simple and fast to implement
  - Only requires storage of the most recent number in the sequence
  - You can generate the same exact sequence with the same seed which is useful when comparing alternative systems/models.

Linear Feedback Shift Registers (LFSRs)

- An LFSR simply combines the bits of a series of registers and shifts the output onto the register.



  - The seed is the initial value of the register.
  - Easy and fast in hardware (1 output bit per clock).
  - Problem: tap configuration can be determined from $2n$ output bits, where n is the length of the LFSR period.

PRNG Vulnerabilities

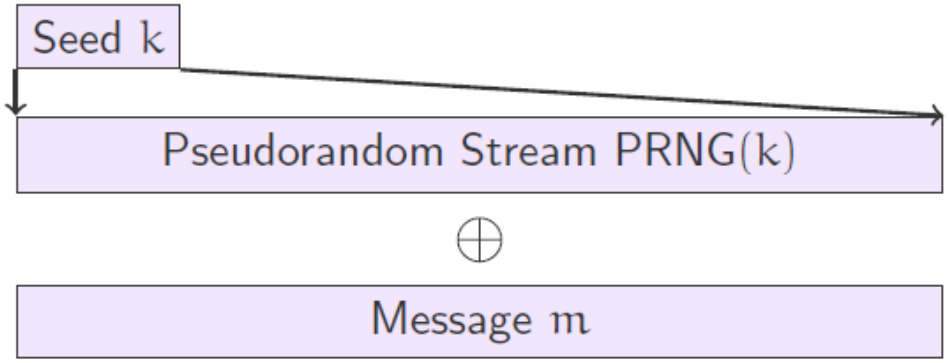
- It is not possible to prove that a sequence of numbers is truly random.
- Be extremely careful with PRNG seeds!
- Hash PRNG inputs with a timestamp or counter
- Reseed the PRNG occasionally
- Use a hash function to protect PRNG outputs if PRNG is susceptible

- Examples of PRNGs:
  - RC4: Based on permutations of a 256-byte array
  - ANSI X9.17: Based on 3DES
  - DSA PRNG: Based on SHA or DES
  - RSAREF PRNG: Based on MD5 hashing and addition modulo 2^{128}
  - Mersenne Twister: Based on the Mersenne prime $2^{19937} - 1$ and is a type of feedback shift register. This is the most widely used general PRNG.

    https://blogs.mathworks.com/cleve/2015/04/17/random-number-generator-mersenne-twister/

PRNGs for Stream Ciphers

- In a one-time-pad, we have a perfectly random r, which is of a size as same as the message m, and the ciphertext is $c = r \oplus m$.
  - Idea: Replace r with a pseudo-random stream.
  - The seed for the PRNG is the key k.
  - Encryption: $E_k(m) = m \oplus \text{PRNG}(k)$.
  - Decryption: $D_k(c) = c \oplus \text{PRNG}(k)$.
- Whenever a key k is expanded to a large pseudorandom stream, this is called a stream cipher.



Stream Ciphers

- Advantages of using a stream cipher:
  - Ease of implementation and use.
  - Secure PRNGs can be a lot faster than block ciphers.
- The security of a stream cipher directly depends on the security of the pseudo-random number generator.
  - It must be computationally hard to find the seed k, or the sequence $\mathrm{PRNG}(k)$.
  - The seed k must be used only once.
  - The PRNG period must be at least as long as the message.
- As with the one-time-pad, stream ciphers by themselves only ensure secrecy. The message may still be modified in transit.

Questions

- What is a symmetric cipher?
- What techniques can be used to break a Vigenere cipher?
- How are PRNGs used in stream ciphers?

THE UNIVERSITY OF SYDNEY