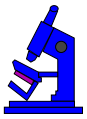


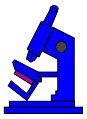

Chapter 2

Lecture 2 - Preliminaries


Note: CORS

You should be getting your tutorial sessions sorted out using CORS!

http://www.cors.nus.edu.sg/



Overheads and notes

You can find all sorts of stuff looking in

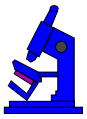
http://www.comp.nus.edu.sg/~cs3235/2003-semesterI/

Question box

If you have any questions, feel free to place them in the question box...

Or stick your hand up...

Or...

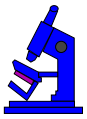


Last session

- ✔ Introduction, setting context
- ✔ Definitions
- ✔ Cæsar cipher, Enigma, Secure shell
- ✔ Insecurity

This session

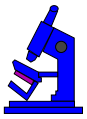
- Finish context
- Math preliminaries
  - XOR
  - Logarithms
  - Fields and groups



This session

- Finish context
- Math preliminaries
  - XOR
  - Logarithms
  - Fields and groups

Diagram for BAG



Safety/control software

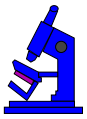
A naive approach to security might involve attempting to ensure that all programs that run on a computer are *safe*, and that all users of computer systems are *trustworthy*.

Checking even one program is a non-trivial task.

The computer *operating system* normally provides some level of software and hardware security for computer systems, combined with some level of user authorization.

Safety/control software

- ✔ **User authorization** means passwords!
- ✔ Systems have grown in complexity over the years.
- ✔ An article shows the changes in the UNIX mechanism



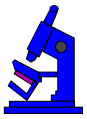
Hardware security

Hardware security in operating systems has been studied in CS2106 (Operating Systems) and other courses. The Kernel/Supervisor bit, processor ring0, memory protection/mapping hardware and so on are all examples of hardware security systems intended to co-operate with the OS to enhance system security.

Software security in operating systems takes many forms. The forms range from ad-hoc changes to operating systems to fix security loopholes as they are found, through to operating systems built from the ground up to be secure.

Example: network security

- ✔ TCP wrappers:
  - ✔ Attacks through poorly controlled TCP or UDP ports.
  - ✔ Wrapper provides single point of control
  - ✔ Default installation disables *all* access
  - ✔ Re-enable on a case-by-case basis.



OS security

NSA have a security-enhanced Linux system:

> *This version of Linux has a strong, flexible mandatory access control architecture incorporated into the major subsystems of the kernel. The system provides a mechanism to enforce the separation of information based on confidentiality and integrity requirements.*

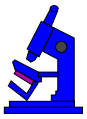
You can read about SELinux at

http://www.nsa.gov/selinux/index.html

OS security

- ✔ Java virtual machine has built-in security model
- ✔ Microsoft point out that the Linux security model is weak...

> *Every member of the Windows NT family since Windows NT 3.5 has been evaluated at either a C2 level under the U.S. Government's evaluation process or at a C2-equivalent level under the British Government's ITSEC process. In contrast, no Linux products are listed on the U.S. Government's evaluated product list.*



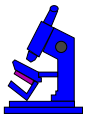
Topic: Assurance

How can we convince ourselves (or our employer) that the computer system is to be trusted?

Building assurance is best done by adopting formal methods to confirm, specify and verify the behaviour of systems.

ITSEC and CC

- ✔ UK, Germany, France, Netherlands produced Information Technology Security Evaluation Criteria (ITSEC).
- ✔ IT Security Evaluation Manual (ITSEM) specifies methodology for evaluation.
- ✔ Common Criteria for Information Technology Security Evaluation is ITSEC, CTCPEC (Canadian Criteria) and US Federal Criteria
- ✔ Accepted by the ISO (ISO15408).



ITSEC

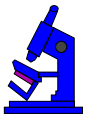
In an article, elements of the first certification of a smart-card system under the European ITSEC level 6 certification are outlined.

This process involved verification of the specification with independent systems, and a formal process for the implementation, deriving it from the specification using the *refinement* process.

Math preliminaries

This chapter and the following chapter are copied verbatim from the "The Laws of Cryptography with Java Code", with permission from Prof Neal Wagner. The book is well worth reading and contains a lot of information that is relevant to this course. You can find the book at

http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf



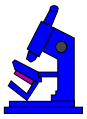
Exclusive-Or

Law XOR-1:
The cryptographer's favorite function is *Exclusive-Or*.

- ✔ Exclusive-Or comes up constantly in cryptography.
- ✔ Same as *addition mod 2*

Exclusive-Or

- ✔ Also as **`xor`** or a plus sign in a circle, $\oplus$.
- ✔ The expression $a \oplus b$ means either a or b *but not both*.
- ✔ Ordinary *inclusive-or* in mathematics means either one or the other *or both*.
- ✔ The exclusive-or function in C / C++ / Java for bit strings as a hat character: **^**.

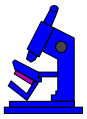


Exclusive-Or for 1-bit

Exclusive-Or		
a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-Or

Message	A	B	C
m	0 1 0 0 0 0 0 1	0 1 0 0 0 0 1 0	0 1 0 0 0 0 1 1 . . .
Key= k	0 0 0 1 0 0 1 1	0 1 1 0 0 1 0 1	0 0 1 1 1 0 0 1 . . .
$K(m) = m \oplus k$	0 1 0 1 0 0 1 0	0 0 1 0 0 1 1 1	0 1 1 1 1 0 1 0 . . .
$K(m)$	R	'	z



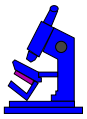
Exclusive-Or

$K(m)$	R	'	z
	0 1 0 1 0 0 1 0	0 0 1 0 0 1 1 1	0 1 1 1 1 0 1 0 . . .
Key= k	0 0 0 1 0 0 1 1	0 1 1 0 0 1 0 1	0 0 1 1 1 0 0 1 . . .
$m = K(m) \oplus k$	0 1 0 0 0 0 0 1	0 1 0 0 0 0 1 0	0 1 0 0 0 0 1 1 . . .
Message	A	B	C

If the bit-stream is random, and not known to an eavesdropper, then this is the most secure system. It is known as a one-time-pad.

Properties of XOR

$a \oplus a = 0$
$a \oplus 0 = a$
$a \oplus 1 = \sim a$, where $\sim$ is bit complement.
$a \oplus b = b \oplus a$ (commutativity)
$a \oplus (b \oplus c) = (a \oplus b) \oplus c$ (associativity)
$a \oplus a \oplus a = a$
if $a \oplus b = c$, then $c \oplus b = a$ and $c \oplus a = b$.



Reminder

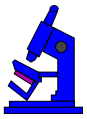
Exchange the values in two variables **a** and **b**

```
temp = a;
a = b;
b = temp;
```


Exchange using XOR

```
a = a xor b;
b = a xor b;
a = a xor b;
```

$$
\begin{aligned}
a' &= a \oplus b \\
b' &= (a \oplus b) \oplus b = a \\
a'' &= (a \oplus b) \oplus a = b
\end{aligned}
$$



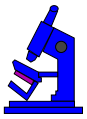
Logarithms

Law LOG-1:
The cryptographer's favorite logarithm is *log base 2*.

- ✔ $y = \log_b x$ is the same as $b^y = x$
- ✔ $b^{(\log_b x)} = x$
- ✔ Logarithm is inverse of exponential.

Logarithms

- ✔ Use logs base 2 in cryptography.
- ✔ $y = \log_2 x$ is the same as $2^y = x$
- ✔ $2^{10} = 1024$ is the same as $\log_2 1024 = 10$.
- ✔ $2^y > 0$ for all y, and
- ✔ $\log_2 x$ is not defined for $x \leq 0$.

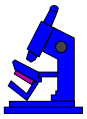


Properties of logs

$\log_2(ab) = \log_2 a + \log_2 b$, for all $a, b > 0$
$\log_2(a/b) = \log_2 a - \log_2 b$, for all $a, b > 0$
$\log_2(1/a) = \log_2(a^{-1}) = -\log_2 a$, for all $a > 0$
$\log_2(a^r) = r \log_2 a$, for all $a > 0, r$
$\log_2(a + b) =$ (Oops! No simple formula for this.)

Examples

Logarithms base 2	
$x = 2^y = 2^{\log_2 x}$	$y = \log_2 x$
$1,073,741,824$	30
$1,048,576$	20
$1,024$	10
8	3
4	2
2	1
1	0



Examples

Logarithms base 2	
$x = 2^y = 2^{\log_2 x}$	$y = \log_2 x$
1	0
$1/2$	-1
$1/4$	-2
$1/8$	-3
$1/1,024$	-10
0	$-\infty$
< 0	undefi ned

Natural logs

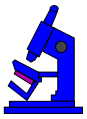
✔ A log base 2 is just a fixed constant times a natural log:

$$\log_2 x = \log_e x / \log_e 2, \text{ (mathematics)}$$

$$= \texttt{Math.log(x)/Math.log(2.0);} \text{ (Java).}$$

✔ The magic constant is:

- ✔ $\log_e 2 = 0.69314\ 71805\ 59945\ 30941\ 72321$, or
- ✔ $1/\log_e 2 = 1.44269\ 50408\ 88963\ 40735\ 99246.$



Proof of formula

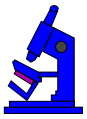
$2^y = x$, or $y = \log_2 x$ (then take $\log_e$ of each side)
$\log_e(2^y) = \log_e x$ (then use properties of logarithms)
$y \log_e 2 = \log_e x$ (then solve for y)
$y = \log_e x / \log_e 2$ (then substitute $\log_2 x$ for y)
$\log_2 x = \log_e x / \log_e 2$.

Bits to represent

Law LOG-2:
The log base 2 of an integer x tells how many bits it takes to represent x in binary.

Thus $\log_2 10000 = 13.28771238$, so it takes 14 bits to represent 10000 in binary. (In fact, $10000_{10} = 10011100010000_2$.) Exact powers of 2 are a special case: $\log_2 1024 = 10$, but it takes 11 bits to represent 1024 in binary, as 10000000000_2.

Similarly, $\log_{10}(x)$ gives the number of decimal digits needed to represent x.



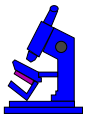
Groups

- A *group* is
  - a set of *group elements* with
  - a *binary operation* f

If one denotes the group operation by $\#$, then the above says that for any group elements a and b, $a\#b$ is defined and is also a group element.

Groups

- Groups
  - are *associative*, meaning that $a\#(b\#c) = (a\#b)\#c$
  - have an *identity element* e satisfying $a\#e = e\#a = a$ for any group element a.
  - have an *inverse* a' any element a satisfying $a\#a' = a'\#a = e$.

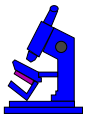


Groups

- ✔ If $a\#b = b\#a$ for all group elements a and b, the group is *commutative*.
- ✔ Otherwise it is *non-commutative*. Notice that even in a non-commutative group, $a\#b = b\#a$ might sometimes be true — for example if a or b is the identity.
- ✔ A group with only finitely many elements is called *finite*; otherwise it is *infinite*.

Examples

- The *integers* (all whole numbers, including 0 and negative numbers) form a group using addition. The identity is 0 and the inverse of a is $-a$.
  - This is an infinite commutative group.
- The *positive rationals* (all positive fractions, including all positive integers) form a group if ordinary multiplication is the operation. The identity is 1 and the inverse of r is $1/r = r^{-1}$.
  - This is another infinite commutative group.



Examples

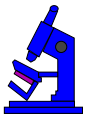
- The *integers mod n* form a group for any integer $n > 0$. This group is often denoted Z_n. Here the elements are 0, 1, 2, $\ldots$, $n-1$ and the operation is addition followed by remainder on division by n. The identity is 0 and the inverse of a is $n-a$ (except for 0 which is its own inverse).
  - This is a finite commutative group.

Non-commutative Group

Consider 2-by-2 non-singular matrices of real numbers (or rationals), where the operation is matrix multiplication: $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Here a, b, c, and d are real numbers (or rationals) and $ad - bc$ must be non-zero. Inverse is

$$\frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

and the identity is $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. This is an infinite non-commutative group.



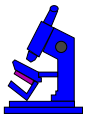
Groups

Law GROUP-1:
The cryptographer's favorite group is the *integers mod n*, Z_n.

In the special case of $n = 10$, the operation of addition in Z_{10} can be defined by $(x + y) \bmod 10$, that is, divide by 10 and take the remainder.

Integers modulo 10

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8



Fields

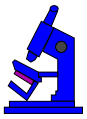
- A field has two operations
  - $+$, with elements of the field forming a commutative group. Identity is 0 and inverse of a is $-a$.
  - $*$, with elements of the field except 0 forming another commutative group, identity denoted by 1 and inverse of a denoted by a^{-1}.

Fields

- There is also the *distributive identity*, linking $+$ and $*$:

$$a * (b + c) = (a * b) + (a * c)$$

- Exclude *divisors of zero*, that is, non-zero elements whose product is zero.
- Equivalent to the following cancellation property: if c is not zero and $a * c = b * c$, then $a = b$.

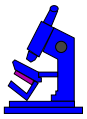


Examples

- ✔ The *rational numbers* (fractions) Q, or the *real numbers* $\mathbb{R}$, or the *complex numbers* $\mathbb{C}$, using ordinary addition and multiplication (extended in the last case to the complex numbers).
  - ✔ These are all infinite fields.

Example: integers mod p

- ✔ The *integers mod p*, denoted Z_p, where p is a prime number ($2, 3, 5, 7, 11, 13, 17, 19, 23, 29, \ldots$).
  - ✔ A group using $+$.
  - ✔ Elements without 0 form a group under $*$.
  - ✔ The identity is clearly 1, but
  - ✔ the inverse of a non-zero element a is not obvious.



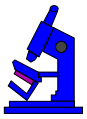
Integers mod p inverse

- In Java, inverse must be x satisfying $(x * a)\%p == 1$.
- Find x using the *extended Euclidean algorithm*:
  - p is prime and a is non-zero, the greatest common divisor of p and a is 1.
  - The extended Euclidean algorithm gives x and y satisfying $x * a + y * p = 1$, or $x * a = 1 - y * p$,
  - and x is the inverse of a.

Field

Law FIELD-1:
The cryptographer's favorite field is the *integers mod p*, denoted Z_p, where *p* is a prime number.

The above field is the only one with p elements. In other words, the field is unique up to renaming its elements, meaning that one can always use a different set of symbols to represent the elements of the field, but it will still be essentially the same.



Another Field

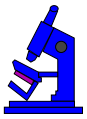
Law FIELD-2:
The cryptographer's *other* favorite field is *GF(2^n)*.

- ✔ A finite field with p^n elements for any integer $n > 1$, denoted $GF(p^n)$.
- ✔ Useful in cryptography with $p = 2$, that is, with 2^n elements for $n > 1$.
- ✔ The case $2^8 = 256$ is used, for example, in the new U.S. Advanced Encryption Standard (AES).

Fermat's Theorem

Law FERMAT-1:
The cryptographer's favorite theorem is Fermat's Theorem.

- ✔ In cryptography, one often wants to raise a number to a power, modulo another number.
- ✔ For the integers mod p where p is a prime (denoted Z_p), there is a result know as Fermat's Theorem, discovered by the 17th century French mathematician Pierre de Fermat, 1601-1665.



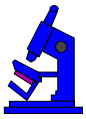
Fermat's theorem

Theorem (Fermat): If p is a prime and a is any non-zero number less than p, then

$$a^{p-1} \bmod p = 1$$

Fermat's theorem

p	a	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}
13	2	2	4	8	3	6	12	11	9	5	10	7	1
13	3	3	9	1	3	9	1	3	9	1	3	9	1
13	4	4	3	12	9	10	1	4	3	12	9	10	1
13	5	5	12	8	1	5	12	8	1	5	12	8	1
13	6	6	10	8	9	2	12	7	3	5	4	11	1
13	7	7	10	5	9	11	12	6	3	8	4	2	1
13	8	8	12	5	1	8	12	5	1	8	12	5	1
13	9	9	3	1	9	3	1	9	3	1	9	3	1
13	10	10	9	12	3	4	1	10	9	12	3	4	1
13	11	11	4	5	3	7	12	2	9	8	10	6	1
13	12	12	1	12	1	12	1	12	1	12	1	12	1



Fermat's theorem

- ✔ For $p = 13$ the value is always 1 by the time the power gets to 12
- ✔ Sometimes the value gets to 1 earlier
- ✔ Lengths of runs are always numbers that divide evenly into 12
- ✔ A value of a for which the whole row is needed is called a *generator*. 2, 6, 7, and 11 are generators.

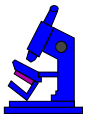
An interesting observation..

Because a to a power mod p always starts repeating after the power reaches $p - 1$, you can do this:

$$a^x \bmod p = a^{x \bmod (p-1)} \bmod p.$$

Thus modulo p in the expression requires modulo $p - 1$ in the exponent. For $p = 13$ as above, then

$$a^{29} \bmod 13 = a^{29 \bmod 12} \bmod 13 = a^5 \bmod 13.$$



Euler

The Swiss mathematician Leonhard Euler (1707-1783) discovered a generalization of Fermat's Theorem which will later be useful in the discussion of the RSA cryptosystem.

Euler's theorem

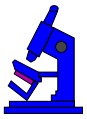
Theorem (Euler): If n is any positive integer and a is any positive integer less than n with no divisors in common with n, then

$$a^{\phi(n)} \bmod n = 1,$$

where $\phi(n)$ is the *Euler phi function*:

$$\phi(n) = n(1 - 1/p_1) \ldots (1 - 1/p_m),$$

and $p_1, \ldots, p_m$ are all the prime numbers that divide evenly into n, including n itself in case it is a prime.



Euler: $n = 15$ and $\phi(n) = 8$

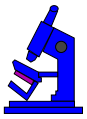
a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}
2	4	8	1	2	4	8	1	2	4	8	1	2	4
3	9	12	6	3	9	12	6	3	9	12	6	3	9
4	1	4	1	4	1	4	1	4	1	4	1	4	1
5	10	5	10	5	10	5	10	5	10	5	10	5	10
6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	4	13	1	7	4	13	1	7	4	13	1	7	4
8	4	2	1	8	4	2	1	8	4	2	1	8	4
9	6	9	6	9	6	9	6	9	6	9	6	9	6
10	10	10	10	10	10	10	10	10	10	10	10	10	10
11	1	11	1	11	1	11	1	11	1	11	1	11	1
12	9	3	6	12	9	3	6	12	9	3	6	12	9
13	4	7	1	13	4	7	1	13	4	7	1	13	4
14	1	14	1	14	1	14	1	14	1	14	1	14	1

Euler's theorem

✔ If n is a prime, then using the formula,

$$\phi(n) = n(1 - 1/n) = n(\frac{n-1}{n}) = n - 1$$

...Euler's result is a special case of Fermat's.



Euler's theorem

✔ Another special case needed for the RSA cryptosystem comes when the modulus is a product of two primes: $n = pq$. Then

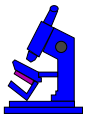
$$\phi(n) = n(1 - 1/p)(1 - 1/q) = (p - 1)(q - 1)$$

Table

Table illustrates Euler's theorem for $n = 15 = 3 \cdot 5$, with

$$\phi(15) = 15 \cdot (1 - 1/3) \cdot (1 - 1/5) = (3 - 1) \cdot (5 - 1) = 8$$

Notice here that a 1 is reached when the power gets to 8 (actually in this simple case when the power gets to 2 or 4), but only for numbers with no divisors in common with 15. For other base numbers, the value never gets to 1.



Euler

Arithmetic in the exponent is taken mod $\phi(n)$, so that, if a has no divisors in common with n,

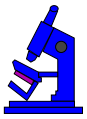
$$a^x \bmod n = a^{x \bmod \phi(n)} \bmod p.$$

If $n = 15$ as above, then $\phi(n) = 8$, and if neither 3 nor 5 divides evenly into a, then $\phi(n) = 8$. Thus for example,

$$a^{28} \bmod 15 = a^{28 \bmod 8} \bmod 15 = a^4 \bmod 15.$$

Summary of topics

In this section, we introduced "Cryptographers favorites"



Further study

- The Laws of Cryptography with Java Code
  http://www.cs.utsa.edu/~wagner/lawsbookcolor/laws.pdf