CS3235
Second lecture

Hugh Anderson

National University of Singapore
School of Computing

August, 2006
Life...

This study suggests that pessimists are a whole lot better at judging the true state of affairs than optimists.

Yeah, but I bet they used a sloppy experimental protocol. I doubt that the conclusions have any validity.
1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
Outline

1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
Note: Tutorials

**Tutorial sessions start next week**

- I will put up the questions on the web site
- If you still have not got a tutorial selected you need to access
  - [https://mysoc.nus.edu.sg/oratut/swap/index.html](https://mysoc.nus.edu.sg/oratut/swap/index.html)
Three messages, locked at all times...

- **RED** – Locked on left-hand-side
- **BLUE** – Locked on right-hand-side

Diagram for lock protocol
Outline

1. Administrivia
2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance
3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
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*.

But *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 - a first step

- **User authorization** means passwords!
- Systems have grown in complexity over the years.
- An article shows the changes in the UNIX mechanism
Various features have been added to computer hardware and software

- **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 management

TCP wrappers:
- Many attacks come through poorly controlled TCP or UDP ports.
- Wrapper provides a single point of control
- The default installation disables *all* access
- And then you re-enable on a case-by-case basis.
Example: 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

Languages and OS security models

Java sand box, and NT security model

- **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.*
Outline

1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
 assurance and trust

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

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

**ITSEC and CC**

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

Hugh Anderson

CS3235 Second lecture
Using the standard to ensure a smart-card system is safe

- In this article, elements of the first certification of a smart-card system under the European ITSEC level 6 certification are outlined. [Woo98]
- 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.
A recent outbreak...

A worm inside a phone

- You can read about CommWarrior at
  
  http://www.f-secure.com/v-descs/commwarrior.shtml
  
- Phone runs a particular version of the Symbian OS and user needs to give permission to install the worm.
- Uses bluetooth and MMS, hence battery concerns.
Outline

1. Administrivia
2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance
3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
The teacher pretended that algebra was a perfectly natural affair, to be taken for granted, whereas I didn’t even know what numbers were. Mathematics classes became sheer terror and torture to me. I was so intimidated by my incomprehension that I did not dare to ask any questions. [Carl Jung]
Exclusive-Or comes up constantly in cryptography.

- Same as *addition mod 2*
- 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*: $\hat{}$. 
Exclusive-Or for 1-bit

A table showing the xor function for one bit

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a ⊕ b</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Exclusive-Or for multiple bits

We can encode messages by using XOR on every bit of the message.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>m</td>
<td>0 1 0 0 0 0 0 1</td>
<td>0 1 0 0 0 0 1 0</td>
<td>0 1 0 0 0 0 1 1</td>
</tr>
<tr>
<td>Key = k</td>
<td>0 0 0 1 0 0 1 1</td>
<td>0 1 1 0 0 1 0 1</td>
<td>0 0 1 1 1 0 0 1</td>
</tr>
<tr>
<td>K(m) = m ⊕ k</td>
<td>0 1 0 1 0 0 1 0</td>
<td>0 0 1 0 0 1 1 1</td>
<td>0 1 1 1 1 0 1 0</td>
</tr>
<tr>
<td>K(m)</td>
<td>R</td>
<td></td>
<td>z</td>
</tr>
</tbody>
</table>

The message is ABC, and the ASCII code is shown.
The encoded message is R’z.
# Exclusive-Or

## The one-time pad

<table>
<thead>
<tr>
<th>$K(m)$</th>
<th>R</th>
<th>$'$</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 1 0 0 1 0</td>
<td>0 0 1 0 0 1 1 1</td>
<td>0 1 1 1 1 0 1 0 . . .</td>
<td></td>
</tr>
<tr>
<td><strong>Key = k</strong></td>
<td>0 0 0 1 0 0 1 1</td>
<td>0 1 1 0 0 1 0 1</td>
<td>0 0 1 1 1 0 0 1 . . .</td>
</tr>
<tr>
<td>$m = K(m) \oplus k$</td>
<td>0 1 0 0 0 0 0 1</td>
<td>0 1 0 0 0 0 1 0</td>
<td>0 1 0 0 0 0 1 1 . . .</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0 0 0 0 1</td>
<td>0 1 0 0 0 0 1 0</td>
<td>0 1 0 0 0 0 1 1 . . .</td>
<td></td>
</tr>
</tbody>
</table>

- If the bit-stream $(k)$ 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 set of rules about the XOR function

\[ a \oplus a = 0 \]
\[ a \oplus 0 = a \]
\[ a \oplus 1 = \sim a, \text{ where } \sim \text{ is bit complement.} \]
\[ a \oplus b = b \oplus a \text{ (commutativity)} \]
\[ a \oplus (b \oplus c) = (a \oplus b) \oplus c \text{ (associativity)} \]
\[ a \oplus a \oplus a = a \]

if \( a \oplus b = c \), then \( c \oplus b = a \) and \( c \oplus a = b \).
How to exchange the values in two variables a and b

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

Note that it uses an extra variable temp.
Exchange using XOR

Exchange the values in two variables \(a\) and \(b\)

\[
\begin{align*}
a &= a \oplus b; \\
b &= a \oplus b; \\
a &= a \oplus b;
\end{align*}
\]

Why does it work? Consider the following:

\[
\begin{align*}
a' &= a \oplus b \\
b' &= (a \oplus b) \oplus b = a \\
a'' &= (a \oplus b) \oplus a = b
\end{align*}
\]
Outline

1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
Logarithms

The logarithm is (kind of) an inverse of an exponential

- \( y = \log_b x \) is the same as \( b^y = x \)
- \( b^{(\log_b x)} = x \)

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 maps (\(\ast,\div\)) to (\(+,-\)) and exponential to \(\ast\):

\[
\log_2(ab) = \log_2 a + \log_2 b, \text{ for all } a, b > 0
\]
\[
\log_2(a/b) = \log_2 a - \log_2 b, \text{ for all } a, b > 0
\]
\[
\log_2(1/a) = \log_2(a^{-1}) = -\log_2 a, \text{ for all } a > 0
\]
\[
\log_2(a^r) = r \log_2 a, \text{ for all } a > 0, r
\]
\[
\log_2(a + b) = \text{(Oops! No simple formula for this.)}
\]
### Logarithms base 2

<table>
<thead>
<tr>
<th>$x = 2^y$</th>
<th>$y = \log_2 x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,073,741,824</td>
<td>30</td>
</tr>
<tr>
<td>1,048,576</td>
<td>20</td>
</tr>
<tr>
<td>1,024</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x = 2^y$</th>
<th>$y = \log_2 x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>–1</td>
</tr>
<tr>
<td>1/4</td>
<td>–2</td>
</tr>
<tr>
<td>1/8</td>
<td>–3</td>
</tr>
<tr>
<td>1/1,024</td>
<td>–10</td>
</tr>
<tr>
<td>0</td>
<td>–∞</td>
</tr>
<tr>
<td>&lt; 0</td>
<td>undefined</td>
</tr>
</tbody>
</table>
Natural logs are logs-to-the-base-e

- A log base 2 is just a fixed constant times a natural log:
  \[ \log_2 x = \frac{\log_e x}{\log_e 2}, \text{ (mathematics)} \]
  \[ = \text{Math.log}(x)/\text{Math.log}(2.0) \text{ (Java)}. \]

- The magic constant is:
  - \[ \log_e 2 = 0.6931471805599453094172321, \text{ or} \]
  - \[ 1/\log_e 2 = 1.4426950408889634073599246. \]
Log-to-the-base-2 computes the number of bits needed to represent a number (... sort of ...)

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 $x$. 
Outline

1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
Steps towards finite fields

We will look at groups, (skip over rings), fields, finite fields...

- Interested in closed algebraic systems/structures
  (Compare +1 with +1 modulo 256)

A group is

- a set of group elements with
- a binary operation $\bullet$

If one denotes the group operation by $\bullet$, then the above says that for any group elements $a$ and $b$, $a \bullet b$ is defined and is also a group element (i.e. it is closed)
Properties of Groups

For all group elements \(a, b, c\), GROUPS are:

- **associative**, meaning that \(a \cdot (b \cdot c) = (a \cdot b) \cdot c\)
- have an **identity** \(e\) satisfying \(a \cdot e = e \cdot a = a\) for any \(a\).
- have an **inverse** \(a'\) satisfying \(a \cdot a' = a' \cdot a = e\).

and

- If \(a \cdot b = b \cdot a\) then the group is **commutative** or **abelian**.
- Otherwise it is **non-commutative**. Notice that even in a non-commutative group, \(a \cdot b = b \cdot a\) 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 of infinite groups

(Integers, +), and (positive rationals, *)

- 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**.
Example of a finite group

(Integers (mod N), + (mod N))

- The *integers mod n* form a group for any integer $n > 0$. This group is often denoted $\mathbb{Z}_n$. 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*. 
Integers modulo 10

A table showing the finite group "Integers modulo 10"

<table>
<thead>
<tr>
<th>+</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
Fields

A field has two operations traditionally called + and *

- +, 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 $a^{-1}$.

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

$$a \ast (b + c) = (a \ast b) + (a \ast c)$$

- If $c$ is not zero and $a \ast c = b \ast c$, then $a = b$. 

Hugh Anderson  
CS3235 Second lecture
Examples of infinite fields

- The **rational numbers** (fractions) $\mathbb{Q}$, 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 of finite field: integers mod p (p is prime)

Integers modulo a prime

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

1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
### Modular arithmetic: $+, \ast$ in $\mathbb{Z}_7$

#### A finite field: $+, \ast$ in $\mathbb{Z}_7$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ast$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Modular arithmetic: $+, \times$ inverses in $\mathbb{Z}_7$

The inverses of each element for the field $(\mathbb{Z}_7, +, \times)$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$-a$</th>
<th>$a^{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
Integers mod $p$ inverse

How to find the inverse?

- Inverse must be $x$ satisfying $(x \ast a) \mod p = 1$.
- 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 $x \ast a + y \ast p = 1$, or $x \ast a = 1 - y \ast p$,
  - and $x$ is the inverse of $a$. 
Properties of finite fields

Fields are unique

1. Fields are unique up to renaming of their 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.

2. Their size must be $p^n$ where $p$ is a prime and $n$ a positive integer.
### Modular arithmetic: $+, \ast$ in $\mathbb{Z}_8$

#### Table: Modular arithmetic in $\mathbb{Z}_8$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>+</strong></td>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

#### Table: Modular arithmetic in $\mathbb{Z}_8$ (Multiplication)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$\ast$</strong></td>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Modular arithmetic: $+, \ast$ inverses in $\mathbb{Z}_8$

<table>
<thead>
<tr>
<th>$a$</th>
<th>$-a$</th>
<th>$a^{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>
Modular arithmetic in $\mathbb{Z}_8$

**Something is wrong with $\mathbb{Z}_8$!**

- Note that in $\mathbb{Z}_8$, modular arithmetic using ordinary addition and multiplication does not form a field.
- Why not? ... Well - because there are not multiplicative inverses....

**But...But...**

8 = $2^3$, so we can have a field of size 8 can’t we?

- Yes - but the operations are not *simple* arithmetic $+$ and $\times$.
- Instead we use *polynomial* arithmetic (to be explained...)
Another field: $GF(2^n)$

Using different operations we CAN have finite fields of size $p^n$

- 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).
- Arithmetic different...
Outline

1. Administrivia

2. The topics... (Continued)
   - Safety/control/hardware/software
   - Steps towards assurance

3. Mathematics
   - Exclusive-Or
   - Logarithms
   - Groups, fields, finite fields
   - Modular arithmetic
   - Polynomial arithmetic
Polynomial arithmetic:

\[3x^5 + x^2 + x + 1.\]

**Whats a polynomial??**

- A polynomial of degree \( n \) is \( \sum_{i=0}^{n} a_i x^i \).
- Consider the set where the coefficients \( a_i \) belong to \( \mathbb{Z}_p \).
- Where \( p = 2 \), the coefficients are either 0 or 1.
- In this case, polynomial addition is xor over the coefficients.
Consider the addition of 3 and 6:

<table>
<thead>
<tr>
<th>3 = 011</th>
<th>x + 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊕ 6 = 110</td>
<td>x² + x</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>5 = 101</td>
<td>x² + 1</td>
</tr>
</tbody>
</table>

Coefficients may be only 0 or 1, and we do the arithmetic modulo 2.

For multiplication, if it results in too large a polynomial, then we reduce it, by dividing it by an irreducible polynomial of degree \( n \), and take the residue.
The multiplication has resulted in a polynomial that is too large, so we reduce it.
Reduction

Multiplication of 7 by 7: reduction

\[
\begin{array}{c|c}
1011 & 10101 \\
\hline
1011 & 1011 \\
11 & \\
\end{array}
\]

\[
x^3 + x + 1
\]

Finally, we have that in \( GF(2^3) \), \( 7 \times 7 = 3 \).
Addition, multiplication in: $GF(2^3)$

### The table for $GF(2^3)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
Inverses in \( GF(2^3) \)

<table>
<thead>
<tr>
<th>( a )</th>
<th>(-a)</th>
<th>( a^{-1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>