Midterm review

ECE 422: Reliable and Secure Systems Design

UNIVERSITY OF ALBERTA

Instructor: An Ran Chen
Term: 2024 Winter

What is the format of the exam?

- Classroom: ETLC E2-002
- Duration: 45 minutes
  - 30 minutes should be enough
  - Please arrive 10 minutes before
- Closed book
- Materials: all the materials posted in the lecture slides
  - There will be a question on the project
- The midterm counts for 25% of the overall course grade

What types of question to expect?

- True/False
- Multiple choice
- Short answer
- Computational questions
    - Please bring a calculator

What to expect on the midterm?

- Concepts: things to know
  - Understand the definition
  - True/false or multiple choice questions
- Explanation: things to explain
  - Be able to explain the concepts, and give examples
  - Multiple choice questions or short answer questions
- Problem: things to solve
  - Understand the mechanism behind the concepts
  - Computational questions

List of materials to prepare you

- Consider the review slides a study guide and a sample midterm
  - Computational questions available
- Go through the concepts in the review slides
- Go through the questions in the lecture slides
- Textbooks: Google Drive link
  - Building Secure and Reliable Systems
- Materials from the past years: Google Drive link
  - Software Redundancy -> Lecture 4 Fault-Tolerant Design Winter 2024
  - Sample midterm available only for the format of the exam

Software development methodologies

- Lecture 1 and 2: DevOps
  - Agile methodology
    - Workflow
    - User stories
    - Planning poker
  - Continuous integration, continuous delivery, continuous deployment
  - Docker container

Software development methodologies

- Lecture 3: Software Reliability Engineering
    - Availability, reliability [Explanation]
        - Mean time between failures, mean time to repair
    - Service level agreement, objective, indicator (SLA, SLO, SLI)
    - Error budget

Availability

MTBF (Mean Time Between Failure) is the average time between two consecutive failures.

$$\text{MTBF} = \frac{\text{Operational Time}}{\text{Number of Failures}}$$

MTTR (Mean Time to Repair) is the average time it takes to restore the system after a failure

$$\text{MTTR} = \frac{\text{Total Repair Time}}{\text{Number of Failures}}$$

Availability is the percentage of time that a system is operational

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTBF+MTTR}}$$

Reliability

Reliability is the likelihood that a system will perform its function without failure.

- Calculated by MTBF (Mean Time Between Failure)
- The higher MTBF, the more reliable and available the system becomes

How to define reliability standards?

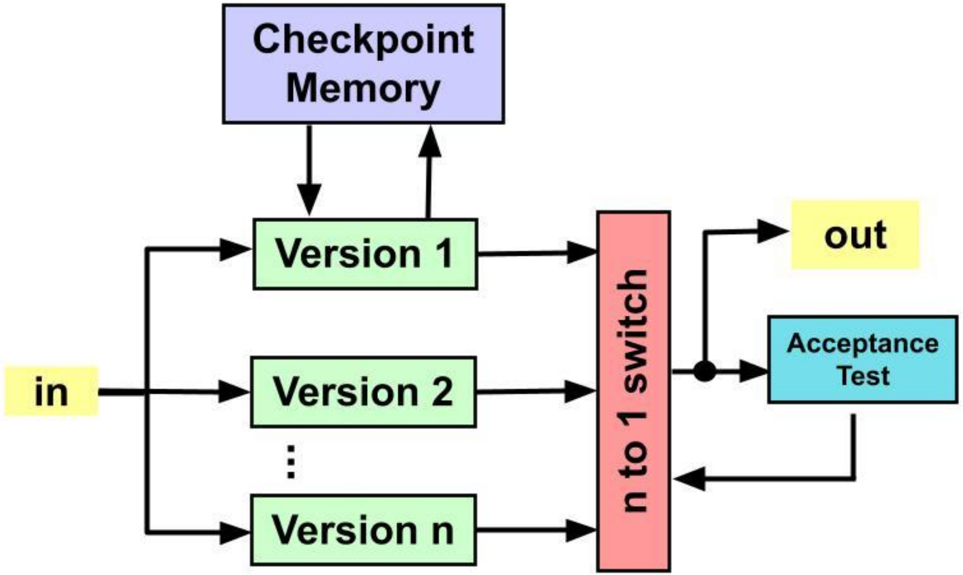
- Risks
  - Safety-critical systems vs personal websites
- Customer expectations
  - Translating “it should never fail” into “99.99% chance of successful operation”
  - Function, environment, and probability of failure
    - E.g., Sales transaction should work 99.99% of the time during Black Friday sales
- SLAs (Service level agreements)

Reliability designs

- Lecture 4: Fault recovery
  - Backward error recovery
  - Forward error recovery
- Lecture 4: Fault tolerance techniques
  - Exception handling
  - Recovery blocks [Explanation]
  - N-version programming [Explanation]
  - N self-checking programming [Explanation]

Recovery blocks

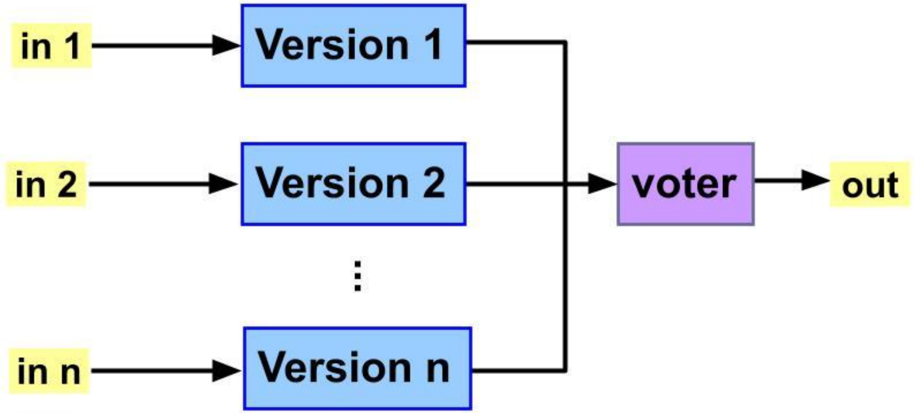
Example



1. Run the primary version for the acceptance test
2. If the primary version fails, roll back the state of the system before the execution
3. Run the next version for the same acceptance test until there is an acceptable output
4. If none produce acceptable outputs, the system fails

N-version programming

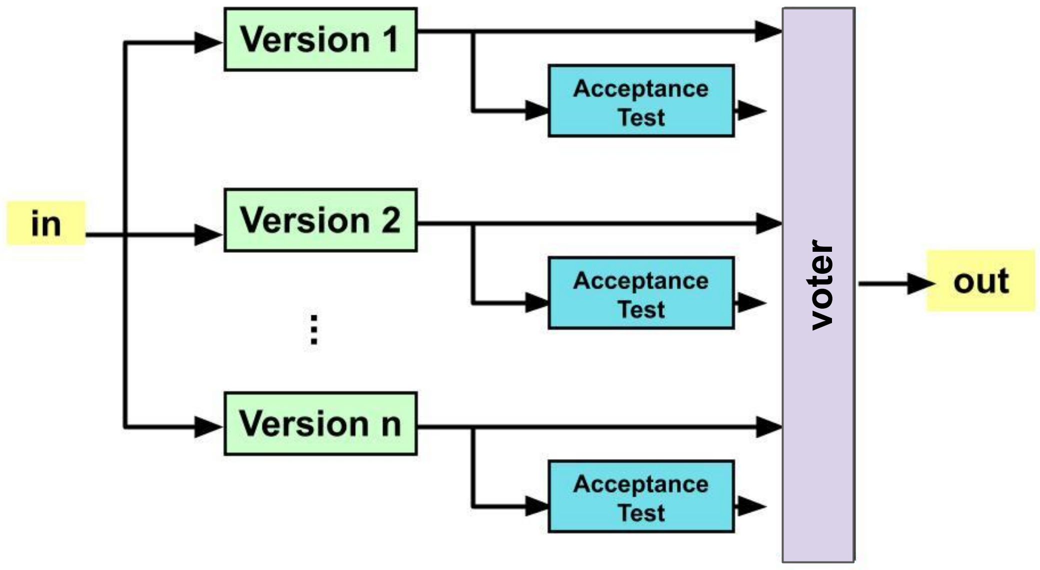
Example



1. Run all versions concurrently
2. Run the decision mechanism based on the voter (e.g., majority win)
3. Return the most common output from the individual versions

N self-checking programming

Example: N self-checking by acceptance tests



1. Run all versions concurrently or sequentially
2. Run the corresponding acceptance test for each version
3. Run the voter
4. Return the majority agreement as output (only applied to versions that have passed their acceptance test)

Reliability designs

- Lecture 5: Fault removal
  - Functional testing
    - Unit test, integration test, acceptance test, regression test
  - Structural testing
    - Mutation test, data flow test, control flow test
    - Code coverage [Explanation]
      - Statement, branch, path coverage
- Lecture 5: Dependability
  - Impairments, measures, means

Code coverage

```
x = 0;
if (condition)
   x = x + 1;
y = 10/x;
```

Test 1 where condition = true

- 100% statement coverage
- No error found in the code

Test 2 where condition = false

- 75% statement coverage
- Error found in the code

Take-home 1: there is an insensitivity of statement coverage to control structures.

Take-home 2: 100% statement coverage does not mean there is no bug in the code.

Code coverage

```
if (condition1)
   x = 0;
else
   x = 2;
if (condition2)
   y = 10*x;
else
   y = 10/x;
```

Test 1 where condition1 = true, and condition2 = true,

Test 2 where condition1 = false, condition2 = false,

- 100% branch coverage
- No error found in the code

Test 3 where condition1 = true,

and condition2 = false,

- 50% branch coverage
- Error found in the code

Take-home 1: 100% branch coverage does not mean there is not bug in the code.

Branch coverage = (Number of Decisions Outcomes tested / Total Number of Decision Outcomes) x 100 %

Reliability designs

- Lecture 6: Fault localization
  - Rubber duck debugging
  - Spectrum-based fault localization
    - 4-step process [Explanation]
    - Suspiciousness score [Problem]
      - Ochiai formula given
  - Information retrieval-based fault localization
    - 3-step process [Explanation]
    - Suspiciousness score [Problem]
      - Cosine similarity formula given

Spectrum-based fault localization

Question: Based on the following execution profile, find the most suspicious statement and compute its suspiciousness score.

	T_1	T_2	T_3
S_1	✓		
S_2	✓		
S_3			✓
S_4	✓	✓	
Result	P	F	F

$$Ochiai(element) = \frac{e_f}{\sqrt{(e_f + n_f) \cdot (e_f + e_p)}}$$

e_f	Number of failed tests that execute the program element.
e_p	Number of passed tests that execute the program element.
n_f	Number of failed tests that do not execute the program element.
n_p	Number of passed tests that do not execute the program element.

Spectrum-based fault localization

Question: Based on the following execution profile, find the most suspicious statement and compute its suspiciousness score.

	T_1	T_2	T_3
S_1	✓		
S_2	✓		
S_3			✓
S_4	✓	✓	
Result	P	F	F

Solution: S_3 is the most suspicious with a score of 0.71

- S_1 = 0, no failed test
- S_2 = 0, no failed test
- S_3 = 0.71, $e_f = 1$, $n_f = 1$
- S_4 = 0.50, $e_f = 1$, $e_p = 1$, $n_f = 1$

$$Ochiai(element) = \frac{e_f}{\sqrt{(e_f + n_f) \cdot (e_f + e_p)}}$$

e_f	Number of failed tests that execute the program element.
e_p	Number of passed tests that execute the program element.
n_f	Number of failed tests that do not execute the program element.
n_p	Number of passed tests that do not execute the program element.

Information retrieval-based fault localization

Question: Based on the following bug report and source file, what is the suspiciousness score of the file?

- Bug report = “a problem with the classNotFound exception.”
- Source file = “get classNotFound exception return exception”

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Information retrieval-based fault localization

Step 1: create a vector representation of the query and document.

- Bug report/query = “a problem with the classNotFound exception.”
- Source file/document = “ get classNotFound exception return exception”

	a	problem	with	the	classNotFound	exception	get	return
A	1	1	1	1	1	1	0	0
B	0	0	0	0	1	2	1	1

Vector representation:

- A = [1, 1, 1, 1, 1, 1, 0, 0]
- B = [0, 0, 0, 0, 1, 2, 1, 1]

Information retrieval-based fault localization

Step 2: calculate the dot product and magnitude of these vectors

Vector representation:

- A = [1, 1, 1, 1, 1, 1, 0, 0]
- B = [0, 0, 0, 0, 1, 2, 1, 1]

Dot product of the vectors:

$A * B = 1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 0 + 1 \times 1 + 1 \times 2 + 0 \times 1 + 0 \times 1 = 3$

Magnitude of the vectors:

$|| A || = \sqrt{(1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 0 + 0)} = \sqrt{6}$

$|| B || = \sqrt{(0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 2^2 + 1^2 + 1^2)} = \sqrt{5}$

Information retrieval-based fault localization

Step 3: calculate the cosine similarity

$$similarity(A, B) = \frac{A * B}{\|A\| \, \|B\|} = \frac{3}{\sqrt{6} * \sqrt{5}} = 0.5477$$

The bug report and source code file could be said to be 55% similar.

Information redundancy

- Lecture 7: Error detecting and correcting code
  - Code, codeword, word, codespace
  - Encoding and decoding
  - Hamming distance
  - Code distance in detection and correction [Explanation]
  - Information rate, formula given (in Lecture 8)
  - Repetition codes
  - Parity codes

Code distance

Question: Give the appropriate (n,k,d) designation for a (7, 4) Hamming code. Also give the number parity bits and the information rate.

Hint 1: n is number of bits in each codeword, k is the number of data bits, and d is the minimum Hamming distance between codewords.

Hint 2: a (7, 4) Hamming code can correct and detect any single-bit error.

Information rate = k/n

Code distance

Question: Give the appropriate (n,k,d) designation for a (7, 4) Hamming code. Also give the number parity bits and the information rate.

Solution: (7, 4) Hamming code uses 7 bits to encode 4 bits of data

- n = 7, codeword bits
- k = 4, data bits
- d = 3, minimal Hamming distance
    - To correct at least one (1) bit error, the code distance must be larger or equal to 2(1)+1
- parity bits = 7 - 4 = 3
- Information rate = 4/7

Information rate = k/n

Information redundancy

- Lecture 8: Hamming codes
  - Encoding and decoding [Problem]
    - Page 16 and 21 on (15, 11) Hamming code
  - Error detection and correction
  - Extended Hamming codes [Problem]

Error detection

data → encoding → codeword

-			0
	1	1	1
	0	1	0
0	1	0	0

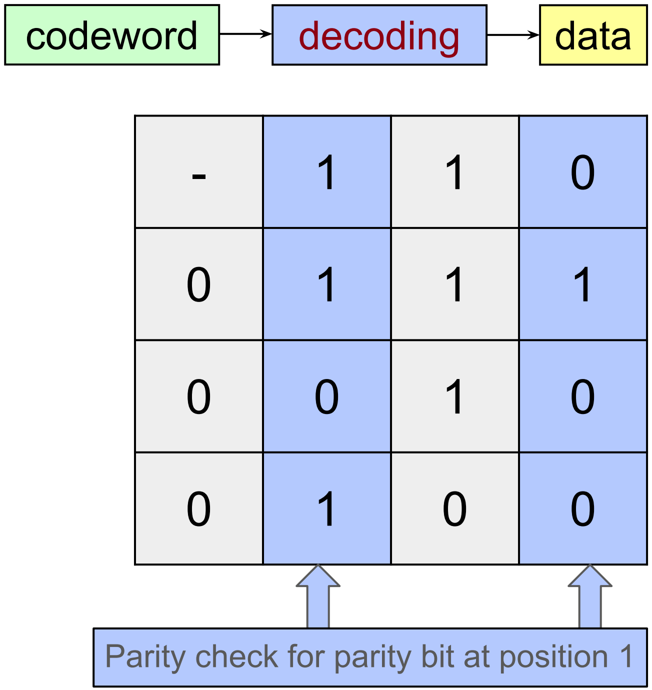
Example: using even parity

Data: {0 1 1 1 0 1 0 0 1 0 0}

Step 1: calculate parity bit #1 at position 1

- We check the parity of the bits in the second and last columns.
- The bits {0 1 1 0 0 1 0} contain three “1”s.
- That is an odd number of “1”s.
- Therefore, we set the first parity bit to 1 to make up for an even parity.

Error detection



Example: using even parity

Data: {0 1 1 1 0 1 0 0 1 0 0}

Scenario 1: No error

The receiver repeat the same process.

Parity bit at position 1: {1 0 1 1 0 0 1 0}, even parity, no error

Parity bit at position 2: {1 0 1 1 1 0 0 0}, even parity, no error

Parity bit at position 4: {0 1 1 1 0 1 0 0}, even parity, no error

Parity bit at position 8: {0 0 1 0 0 1 0 0}, even parity, no error

Information redundancy

- Lecture 9 and 10: Cyclic codes
    - Linear and cyclic properties [Explanation]
    - Encoding and decoding [Problem]
        - Polynomial multiplication and division
    - Error detection

Encoding

Suppose $g(x) = (1+x+x^3)$ for a (7,4) cyclic code

Question: Find the codeword polynomial for the data 1010. (show your steps)

Encoding

Suppose $g(x) = (1+x+x^3)$ for a (7,4) cyclic code

Question: Find the codeword polynomial for the data 1010. (show your steps)

Thought process: We have the values g(x) and d(x), asked to calculate c(x)

- Solve c(x) = d(x)g(x) using the cyclic property
  - Step 1: write down the data as a 7-bit codeword (n = 7)
  - Step 2: apply the cyclic property to calculate polynomial multiplication
  - Step 3: convert the codeword into a codeword polynomial

Encoding

Suppose $g(x) = (1+x+x^3)$ for a (7,4) cyclic code

Question: Find the codeword polynomial for the data 1010. (show your steps)

Solution: Solve c(x) = d(x)g(x) using the cyclic property

Step 1: write down the data as a 7-bit codeword

- data = {1010000}

Step 2: apply the cyclic property: shift the codeword based on the power of x

- d(x)·g(x) = 1010000 + 0101000 + 0001010 = 1110010

Step 3: convert the codeword into a codeword polynomial

- $1110010 = 1 + x + x^2 + x^5$

Decoding

Suppose *g(x) = (1101)* for a (7,4) cyclic code

Question: Bob receives a codeword 0110111 from Alice. Is there an error? If yes, justify why. (show your steps)

Decoding

Suppose *g(x) = (1101)* for a (7,4) cyclic code

Question: Bob receives a codeword 0110111 from Alice. Is there an error? If yes, justify why. (show your steps)

Thought process: We have the values g(x) and c(x), asked whether there is a remainder in c(x)/g(x). If yes, then there is a error.

- Solve d(x) = c(x)/g(x)
  - Step 1: convert c(x) and g(x) into polynomials
  - Step 2: calculate polynomial division c(x)/g(x), check for remainder

Decoding

Suppose *g(x) = (1101)* for a (7,4) cyclic code

Question: Bob receives a codeword 0110111 from Alice. Is there an error? If yes, justify why. (show your steps)

Solution: Solve d(x) = c(x)/g(x)

Step 1: convert c(x) and g(x) into polynomials

- $g(x) = 1 + x + x^3$
- $c(x) = x + x^2 + x^4 + x^5 + x^6$

Decoding

Solution (cont.):

Step 2: calculate polynomial division c(x)/g(x), check for remainder

- $(x + x^2 + x^4 + x^5 + x^6)/(1 + x + x^3) = x^2 + x^3$ remainder x

$$
\begin{array}{l|l}
x^6 + x^5 + x^4 + x^2 + x & x^3 + x + 1 \\
\cline{2-2}
x^6 + x^4 + x^3 & x^3 + x^2 \\
\hline
\quad x^5 + x^3 + x^2 + x & \\
\quad x^5 + x^3 + x^2 & \\
\cline{1-1}
\qquad\qquad x &
\end{array}
$$

There is a remainder, so the error is detected.

Byzantine fault tolerance

- Lecture 11 and 12: Byzantine fault tolerance
  - Timing failure, omission failure, crash failure, Byzantine failure
  - Fail-stop, fail-noisy, fail-silent, fail-safe, fail-arbitrary
  - The Two Generals Problem
  - The Byzantine Generals Problem [Explanation]
    - E.g., m = 1, n = 3, is it solvable? Why?

System security

- Lecture 13 and 14: CIA triad and digital signature
    - Confidentiality, integrity and availability
    - Hash function
        - Hash collision [Explanation]
    - Digital signature [Explanation]
        - Collision attack
        - SHA256

System security

- Lecture 15: Authentication
  - Authentication methods
    - Password-based, magic links, SMS-based, authenticator apps, biometric authentication
    - Time-based one-time password [Explanation]
    - HMAC-based one-time password [Explanation]
  - Multi-factor authentication

System security

- Lecture 16: Access control
    - Threats, vulnerabilities, attacks
    - Access control lists
    - Models of access controls [Explanation]
        - Discretionary access control (DAC)
            - E.g., what, who, benefits and problems
        - Role-based access control (RBAC)
            - …
        - Mandatory access control (MAC)
            - …
        - Attribute-based access control (ABAC)
            - …

System security

- Lecture 17: Encryption
    - Symmetric encryption [Explanation]
    - Man-in-the-middle attack [Explanation]
    - Asymmetric encryption
        - RSA algorithm [Problem]

RSA algorithm

Alice wants to send a message (m = 3) to Bob through the RSA algorithm. Assume that the two prime numbers used to generate the keys are p = 13, q = 7, and Alice must choose a value e < 10.

Question: What is the ciphertext? (show your calculations and assumptions)

Encryption key (e, n)

- $m^e \mod(n) = c$

RSA algorithm

Alice wants to send a message ($m = 3$) to Bob through the RSA algorithm. Assume that the two prime numbers used to generate the keys are $p = 13$, $q = 7$, and Alice must choose a value $e < 10$.

Question: What is the ciphertext? (show your calculations and assumptions)

Thought process: We have the values m, p, q, asked to calculate c

- We need to calculate the public key (e, n) to find c
    - Step 1: calculate n
    - Step 2: calculate $\varphi(n)$
    - Step 3: find e that satisfies the conditions
    - Step 4: calculate c with (e, n)

RSA algorithm

Solution: We need to calculate the public key (e, n) to find c

Step 1: calculate n

- If p = 13, q = 7, then n = pq = 91

Step 2: calculate φ(n)

- φ(n) = (p-1)(q-1) = 12 x 6 = 72

RSA algorithm

Solution (cont.):

Step 3: find e that satisfies the conditions

- Condition: $1 < e < \varphi(n)$, and given that $e < 10$
  - $1 < e < 10$
- Condition: e and $\varphi(n)$ must be coprime
  - e and 72 must be coprime
  - e cannot be a divisor of 72 including 2, 3, 4, 6, 8, 9
- e can either be 5 or 7
- Public key: (5, 91) or (7, 91)

RSA algorithm

Solution (cont.):

Step 4: calculate c with (e, n)

If e = 5

- $m^e \mod(n) = c$
- $3^5 \mod 91 = 243 \mod 91 = 61$

If e = 7

- $3^7 \mod 91 = 2187 \mod 91 = 3$

Encryption key (e, n)

- $m^e \mod(n) = c$

RSA algorithm

Part I: Bob's public and private key setup

- Chooses two prime numbers, p and q
- Calculate the product $n = pq$
- Solve $\varphi(n) = (p-1)(q-1)$
- Choose numbers e and d so that ed has a remainder of 1 when divided by $\varphi(n)$
  - *$1 < e < \varphi(n)$, where e must be an integer*
  - *e and $\varphi(n)$ must be coprime*
  - e*d (mod φ(n)) = 1

Example

- $p = 11$, $q = 3$
- $n = pq = 33$
- $\varphi(n) = 10 \times 2 = 20$
- Pick e and d so that ed = 20+1

  e.g.,: $e = 3$, $d = 7$
  - 1 < 3 < 20
  - 3 and 7 are coprime

RSA algorithm

Assume that the values of p and q are larger than 5, and the value of n is less than 100.

Question: Alice sends a ciphertext (c = 3) using the public key (5, 91). How can Eve break the ciphertext? (show your calculations and assumptions)

Decryption key (d, n)

- $c^d \bmod(n) = m$

This question also shows why we should use larger prime numbers.

RSA algorithm

Assume that the values of p and q are larger than 5, and the value of n is less than 100.

Question: Alice sends a ciphertext (c = 3) using the public key (5, 91). How can Eve break the ciphertext? (show your calculations and assumptions)

Thought process: We have the values c, e, n, asked to calculate m

- We need to calculate the private key (d, n) to find m
  - Step 1: find p and q
  - Step 2: calculate φ(n)
  - Step 3: find d that satisfies the condition
  - Step 4: calculate m with (d, n)

RSA algorithm

Solution:

Step 1: find p and q

- $p > 5$ and $q > 5$
- If $n = 91$, then $p = 7$, $q = 13$
  - Trial and error, also the only two prime numbers whose product is 91

Step 2: calculate $\varphi(n)$

- $\varphi(n) = (p-1)(q-1) = 6 \times 12 = 72$

RSA algorithm

Solution (cont.):

Step 3: find d that satisfies the conditions

- Condition: $e*d \ (\mathrm{mod}\ \varphi(n)) = 1$
    - Find a value of d so that $5d\ (mod\ 72) = 1$ holds
    - Trial and error is enough to solve this
        - (72+1) / 5 not an integer
        - (144+1) / 5 = 29
        - 29 < 100
- d = 29
- Private key: (29, 91)

RSA algorithm

Solution (cont.):

Step 4: calculate m with (d, n)

- $c^d \bmod(n) = m$
- $61^{29} \bmod 91 = 3$
  - If your calculator does not support the modulus operator
  - Step 1) Break down the exponent, e.g., 29 = 5 + 5 + 5 + 5 + 5 + 4
  - Step 2) Solve for $(61^5)^5 \pmod{91} * 61^4 \pmod{91}$
    - $61^5 \bmod 91 = 3$
    - $61^4 \bmod 91 = 9$
  - Step 3) $3^5 * 9 \pmod{91} = 3$

Decryption key (d, n)

- $c^d \bmod(n) = m$