

Managing Information (CSCU9T4) Data Security

Nadarajen Veerapen
nve@cs.stir.ac.uk



HOME → YOUR PRACTICE → PRACTICE TOPICS → IT

Identifiable patient data lost on three occasions since 2009 by Government information centre

9 April 2014 | By [Alex Matthews-King](#)

Print



Save



Comments (3)



SHARE ON FACEBOOK



SHARE ON TWITTER



EMAIL TO A FRIEND

Patient data security has been breached in four of the last five years, resulting in identifiable records being disclosed by the Government's official information centre without authorisation or to the wrong recipient, a Freedom of Information request has revealed.

The security for hospital inpatient data - including medical information, alongside patient age and postcode - was breached every year from 2009 to 2012.

'Don't damage encryption' rights coalition tells global governments

ENCRYPTION / 11 JANUARY 16 / by MATT BURGESS



20 shares
0 comments

Campaigners from more than 42 countries have called for governments to stop what they see as attacks on **encrypted** communications.

More than 190 civil rights groups and companies - including Amnesty International, Open Rights Group, and the Electronic Frontier Foundation -- said that "backdoors" to access secure communications should not be introduced, and say that doing so in just one place would damage security around the world.



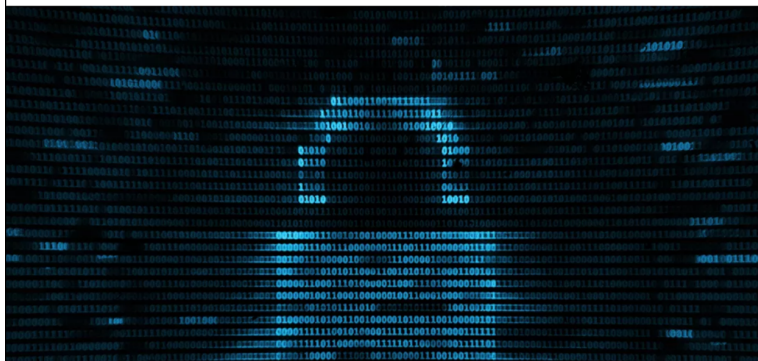
FBI director James Comey has called for an end to "default" encryption *Chip Somodevilla/Getty*

Hackers accessed more personal data from Equifax than previously disclosed

Data included tax identification numbers, e-mail addresses, and additional drivers license information

By [Andrew Liptak](#) | [@AndrewLiptak](#) | Feb 11, 2018, 10:55am EST

[f](#) [t](#) [s](#) SHARE



end to "default"

More than 15,600 bank account numbers and sort codes were stolen, the company said.

Hackers accessed more personal data

3

NEWS SECURITY

Every NHS trust assessed for cyber security has failed

6TH FEBRUARY 2018

Each of the 200 NHS trusts tested for cyber security resilience has failed the test, MPs were told yesterday.

Speaking to the Public Accounts Committee, NHS Digital's Rob Shaw said that while some trusts are close to satisfying the requirements, others have a considerable amount of work still to do.

"The amount of effort it takes for NHS providers in such a complex estate to reach the cyber essential plus standard that we assess against is quite a high bar," he said. "Some of them have failed purely on patching, which is what the vulnerability was around Wannacry."

The assessments began before the WannaCry attack paralysed parts of the NHS and thousands of other organisations in May last year. A further 36 trusts are yet to be assessed.

ers

end to "default"

Hackers accessed more personal data

3

NEWS SECURITY

BBC



Home

News

Sport

Weather

More



NEWS



Technology

Hackers hijack government websites to mine crypto-cash

21 minutes ago



The Information Commissioner's Office (ICO) took down its website after a warning that hackers were taking control of visitors' computers to mine cryptocurrency.

cyber

ers

s failed the

law said
thers have

ex estate
s quite a
which is

erts of the
ther 36

end to "default"

These 2 lectures

What are these lectures about?

- ▶ General concepts of computer security
- ▶ Some practical examples of vulnerabilities and how to fix them in Java.
- ▶ Introduction to cryptography and how to do it in Java.

Resources

- ▶ Charles P. Pfleeger, Shari L. Pfleeger: Security in Computing. 4th Ed. Prentice Hall Professional, 2006
 - ▶ Available at the library as an e-book
- ▶ SEI CERT Oracle Coding Standard for Java
<https://www.securecoding.cert.org/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java>

Slides partly based on past lectures by Jingpeng Li.

General Picture

- ▶ Computer security, also known as cybersecurity or IT security, is the protection of computer systems from the theft or damage to the hardware, software or the information on them, as well as from disruption or misdirection of the services they provide.

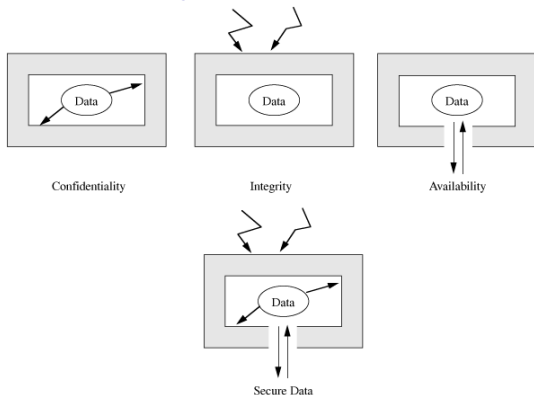
General Picture

- ▶ Computer security, also known as cybersecurity or IT security, is the protection of computer systems from the theft or damage to the hardware, software or the information on them, as well as from disruption or misdirection of the services they provide.
- ▶ Data security means protecting data, such as a database, from destructive forces and from the unwanted actions of unauthorized users.

General Picture

- ▶ Computer security, also known as cybersecurity or IT security, is the protection of computer systems from the theft or damage to the hardware, software or the information on them, as well as from disruption or misdirection of the services they provide.
- ▶ Data security means protecting data, such as a database, from destructive forces and from the unwanted actions of unauthorized users.
- ▶ Note: Wikipedia's entry on computer security gives a good overview, the one on data security is a bit sparse.

Goals of Data Security

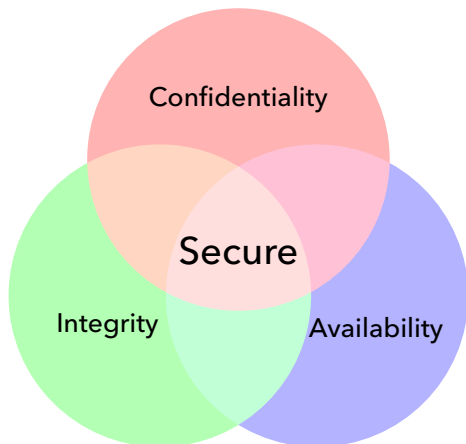


Confidentiality is about preventing unauthorised disclosure of data.

Integrity is about preventing unauthorised modification or corruption of data.

Availability is about preventing the denial of authorised access.

Challenge



Challenge: finding the right balance among the goals, which often conflict.

E.g. We can preserve an asset's confidentiality by preventing everyone from reading it. But then it is not available!

These three characteristics can be independent, can overlap, and can even be mutually exclusive.

Availability

Authentication

- ▶ Identifying an individual as being genuine and not an imposter
- ▶ Achieved by checking something a user

knows: PIN, pswd, DOB

has: key, card, uniform

is: face, fingerprint, iris scan

- ▶ Combine two or more for more secure systems

Authorisation

- ▶ Concerned with what a user is allowed to do
- ▶ Computer system levels

User: create/edit files in their local space, cannot install software

Manager: edit files in project workspaces

Administrator: install software, configure printers, manage accounts

Additional Property

- ▶ Confidentiality, Integrity, Availability are related to prevention
- ▶ Another important property of security is **audit**, related to taking actions after an attack.

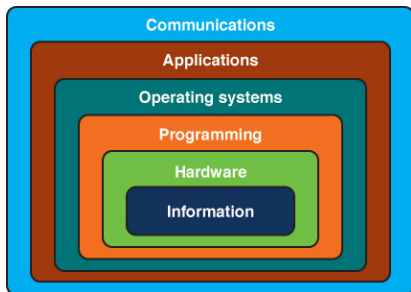
Audit: the ability to conduct a methodological and thorough review of a system

- ▶ Rely on logging/recording actions of system and users
- ▶ May prevent attack and dissuades the attacker due to audit trail left behind (skilled hackers may find ways of covering trails)

Characteristics of Computer Intrusion

A computing system is a collection of hardware, software, data, and people that an organization uses to perform computing tasks.

- ▶ Any part of a computing system can be the target.
 - ▶ Sometimes, we assume that parts of a computing system are not vulnerable to an outsider, but often we are mistaken.
 - ▶ Any system is most vulnerable at its weakest point.



Vulnerabilities, Threats and Controls

- ▶ A computer system has three separate but valuable components: hardware, software and data.
- ▶ A **vulnerability** is a weakness in the security system that might be exploited to cause loss or harm.
- ▶ A **threat** is a possible danger to the system. The danger might be a person (a system cracker or a spy), a thing (a faulty piece of equipment), or an event (a fire or a flood).
- ▶ We use a **control** as a protective measure. A control is an action, device, procedure, or technique that removes/reduces a vulnerability.

Vulnerabilities

Hardware: adding/removing devices, intercepting the traffic to them, or flooding them with traffic until they can no longer function.

Software: replacing, changing or destroying software maliciously or accidentally (e.g., virus, information leaks).

Data: data have a definite value, even though that value is often difficult to measure.

Example 1: confidential data leaked to a competitor (may narrow a competitive edge)

Example 2: flight coordinate data used by an airplane that is guided partly or fully by software (can cost human lives if modified)

Threats

- Interception:** unauthorised access to an asset.
For example, illicit copying of program or data files, wiretapping to obtain data in a network.
- Interruption:** an asset becomes lost, unavailable, or unusable.
For example, malicious destruction of hardware, erasing programs or data files, Denial of Service (DoS) attack
- Modification:** an asset is tampered by an unauthorised party.
For example, changing values in a database, altering programs to perform additional computation.
- Fabrication:** an unauthorised party creates counterfeit objects.
For example, adding records to a database, spurious transactions in a network.

Available Controls

Encryption: 'scrambling' data

Software controls: programs must be secure to prevent outside attack. Internal controls (access limitations), operating systems and network controls, independent control programs (virus scanner, intrusion detection)

Hardware controls: several devices such as smart card encryption, locks or cables, firewalls, devices to verify identity, intrusion detection systems

Policies and procedures: agreed among users, such as regular changes of passwords, training to enforce importance of security

Physical controls: door locks, guards at entry points, backup of software and data

Encryption

- ▶ Take data in their normal, unscrambled state, called **plaintext**, and transform them so that they are unintelligible to the outside observer (**ciphertext**).
- ▶ Addresses the need for **confidentiality** of data.
- ▶ It can be used to ensure **integrity**: data that cannot be read generally cannot easily be changed in a meaningful manner.

More on this in the next lecture.

Some Vulnerabilities caused by Programmers

and how to avoid them

- ▶ It is possible to avoid many vulnerabilities through good coding practices.
- ▶ Your company, or client you are working for, may have internal policies you need to comply with.
- ▶ You can also refer to best practice guides created by trusted external organisations. For example, Carnegie Mellon's Computer Emergency Response Team (CERT) has developed the Oracle Secure Coding Standard for Java:
`https://www.securecoding.cert.org/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java`
- ▶ The aim is "to produce programs that are not only secure, but also safer, more reliable, more robust, and easier to maintain."

Safely reading numbers

(from the previous lecture)

- ▶ Scanner `nextInt` and `nextDouble` can get confused

	2	1	s	t		c	e	n	t	u	r	y	
--	---	---	---	---	--	---	---	---	---	---	---	---	--

- ▶ If the number is not properly formatted, an `InputMismatchException` occurs
- ▶ Use the `hasNextInt` and `hasNextDouble` methods to test your input first

```
if (in.hasNextInt())  
{  
    int value = in.nextInt();    // safe  
}
```

- ▶ They will return true if digits are present
 - ▶ If true, `nextInt` and `nextDouble` will return a value
 - ▶ If not true, they would throw an input mismatch exception

Safely reading numbers

(from the previous lecture)

- ▶ Scanner `nextInt` and `nextDouble` can get confused

	2	1	s	t		c	e	n	t	u	r	y	
--	---	---	---	---	--	---	---	---	---	---	---	---	--

- ▶ If the number is not properly formatted, an `InputMismatchException` occurs
- ▶ Use the `hasNextInt` and `hasNextDouble` methods to test your input first

```
if (in.hasNextInt())  
{  
    int value = in.nextInt();    // safe  
}
```

- ▶ They will return true if digits are present
 - ▶ If true, `nextInt` and `nextDouble` will return a value
 - ▶ If not true, they would throw an input mismatch exception
- ▶ How could an improperly handled error affect the program?

Safely reading numbers

(from the previous lecture)

- ▶ Scanner `nextInt` and `nextDouble` can get confused

	2	1	s	t		c	e	n	t	u	r	y	
--	---	---	---	---	--	---	---	---	---	---	---	---	--

- ▶ If the number is not properly formatted, an `InputMismatchException` occurs
- ▶ Use the `hasNextInt` and `hasNextDouble` methods to test your input first

```
if (in.hasNextInt())  
{  
    int value = in.nextInt();    // safe  
}
```

- ▶ They will return true if digits are present
 - ▶ If true, `nextInt` and `nextDouble` will return a value
 - ▶ If not true, they would throw an input mismatch exception
- ▶ How could an improperly handled error affect the program?
- ▶ Important to ensure that any input, not just numbers, are properly checked and validated.

Divide-by-zero

- ▶ Beyond checking the validity of inputs, it is also important to check that the operations carried out on the data will produce valid results.
- ▶ For example, division and remainder operations are susceptible to divide-by-zero errors (`ArithmeticException`).

```
long num1, num2, result;  
  
/* Initialise num1 and num2 */  
  
if (num2 == 0) {  
    // Handle error  
} else {  
    result = num1 / num2;  
}
```

Integer overflow (1)

- ▶ Be aware of the limitations of the programming language you are using.
- ▶ For example, integer operators in Java do not indicate overflow in any way. See Java language specification:
<http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.2>

Integer overflow (1)

- ▶ Be aware of the limitations of the programming language you are using.
- ▶ For example, integer operators in Java do not indicate overflow in any way. See Java language specification: <http://docs.oracle.com/javase/specs/jls/se7/html/jls-4.html#jls-4.2.2>
- ▶ **Overflow** occurs when a mathematical operation cannot be represented using the supplied integer types. Java's built-in integer operators silently wrap the result without indicating overflow.

```
int maxInt = 2147483647;  
System.out.println("Overflow: " + (maxInt + 1));  
// [OUTPUT] Overflow: -2147483648
```

Type	Inclusive Range
byte	-128 to 127
char	\u0000 to \uffff (0 to 65,535)
int	-2,147,483,648 to 2,147,483,647
long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Integer overflow (2)

```
public static int add(int left, int right) {  
    // May result in overflow  
    return left + right;  
}
```

Integer overflow (2)

```
public static int add(int left, int right) {  
    // May result in overflow  
    return left + right;  
}
```

```
static final int safeAdd(int left, int right) {  
    if (right > 0 ? left > Integer.MAX_VALUE - right  
        : left < Integer.MIN_VALUE - right) {  
        throw new ArithmeticException("Integer overflow");  
    }  
    return left + right;  
}  
  
public static int add(int left, int right) {  
    return safeAdd(left, right);  
}
```

Validate method arguments (1)

- ▶ As a generalisation of the previous slide, validate method arguments to ensure that they fall within the bounds of the method's intended design.
- ▶ **Caller** validation of arguments is done before calling the method.
- ▶ **Callee** validation of arguments is done within the method.

Validate method arguments (1)

- ▶ As a generalisation of the previous slide, validate method arguments to ensure that they fall within the bounds of the method's intended design.
- ▶ **Caller** validation of arguments is done before calling the method.
- ▶ **Callee** validation of arguments is done within the method.
- ▶ Methods that receive arguments across a trust boundary, e.g. all public methods of a library, must perform callee validation of their arguments for safety and security reasons.
- ▶ Other methods, including private methods, should validate arguments that are both untrusted and unvalidated when those arguments may propagate from a public method via its arguments.

Validate method arguments (2)

```
private Object myState = null;

// Sets some internal state in the library
void setState(Object state) {
    myState = state;
}

// Performs some action using the state passed earlier
void useState() {
    // Perform some action here
}
```

Validate method arguments (3)

```
private Object myState = null;

// Sets some internal state in the library
void setState(Object state) {
    if (state == null) {
        // Handle null state
    }

    // Defensive copy here when state is mutable

    if (isInvalidState(state)) {
        // Handle invalid state
    }
    myState = state;
}

// Performs some action using the state passed earlier
void useState() {
    if (myState == null) {
        // Handle no state (e.g., null) condition
    }
    // ...
}
```

Do not suppress or ignore checked exceptions (1)

- ▶ **Checked exceptions** represent invalid conditions in areas outside the immediate control of the program (invalid user input, database problems, network outages, absent files).
- ▶ Programmers often suppress checked exceptions by catching exceptions with an empty or trivial catch block.

Do not suppress or ignore checked exceptions (1)

- ▶ **Checked exceptions** represent invalid conditions in areas outside the immediate control of the program (invalid user input, database problems, network outages, absent files).
- ▶ Programmers often suppress checked exceptions by catching exceptions with an empty or trivial catch block.
- ▶ The `catch` block must either recover from the exceptional condition, rethrow the exception to allow the next nearest enclosing `catch` clause of a `try` statement to recover, or throw an exception that is appropriate to the context of the `catch` block.
- ▶ Suppressing checked exceptions may lead to a problem or an intrusion not being detected, or detected too late.

Do not suppress or ignore checked exceptions (2)

```
try {  
    //...  
} catch (IOException ioe) {  
    ioe.printStackTrace();  
}
```

- ▶ Printing the exception's stack trace can be useful for debugging purposes, but the resulting program execution is equivalent to suppressing the exception.
- ▶ Printing the stack trace can also leak information about the structure and state of the process to an attacker.

Do not suppress or ignore checked exceptions (3)

```
volatile boolean validFlag = false;
do {
    try {
        // If requested file does not exist, throws
        // FileNotFoundException
        // If requested file exists, sets validFlag to true
        validFlag = true;
    } catch (FileNotFoundException e) {
        // Ask the user for a different file name
    }
} while (validFlag != true);
// Use the file
```

Do not allow exceptions to expose sensitive information (1)

- ▶ Information leaks can assist an attacker's efforts to develop further exploits.
- ▶ An attacker may craft input arguments to expose internal structures and mechanisms of the application.
- ▶ Both the exception message text and the type of an exception can leak information.

For example, the `FileNotFoundException` message reveals information about the file system layout, and the exception type reveals the absence of the requested file.

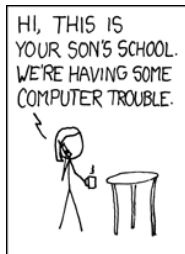
Do not allow exceptions to expose sensitive information (2)

Exception	Description of risk
java.io.FileNotFoundException	Underlying file system structure, user name enumeration
java.sql.SQLException	Database structure, user name enumeration
java.net.BindException	Enumeration of open ports when untrusted client can choose server port
java.util. ConcurrentModificationException	May provide information about thread-unsafe code
javax.naming. InsufficientResourcesException	Insufficient server resources (may aid DoS)
java.util. MissingResourceException	Resource enumeration
java.util.jar.JarException	Underlying file system structure
java.security.acl. NotOwnerException	Owner enumeration
java.lang.OutOfMemoryError	DoS
java.lang.StackOverflowError	DoS

Input Sanitisation

- ▶ Input sanitisation, or validation, is about making sure that input values follow some expected specifications and are safe to use.
- ▶ We have seen some basic examples of how to validate inputs, read specific types, etc. Failure to handle these issues can lead to program crashes, corruption and loss of data.
- ▶ Now we look at some examples of how to actively access and potentially damage data.

Input Sanitisation vs SQL injection (1)



<https://xkcd.com/327/>

OH, DEAR - DID HE
BREAK SOMETHING?

IN A WAY-



DID YOU REALLY
NAME YOUR SON
Robert'); DROP
TABLE Students;-- ?

OH. YES. LITTLE
BOBBY TABLES,
WE CALL HIM.



WELL, WE'VE LOST THIS
YEAR'S STUDENT RECORDS.
I HOPE YOU'RE HAPPY.

AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.



```
String sqlString = "INSERT INTO students (name) VALUES ('"+  
    name + "')";
```

```
Statement stmt = connection.createStatement();
```

```
ResultSet rs = stmt.executeUpdate(sqlString);
```

```
INSERT INTO students (name) VALUES ('Robert');
```

```
DROP TABLE students;--');
```

- ▶ SQL is a language to query and manipulate databases.
- ▶ In SQL, -- indicates a comment.

Input Sanitisation vs SQL injection (2)



<https://xkcd.com/327/>

```
String sqlString = "INSERT INTO students (name) VALUES (?);";
PreparedStatement stmt = connection.prepareStatement(sqlString);
stmt.setString(1, name);
ResultSet rs = stmt.executeUpdate();
```

Prepared statements force the user input to be handled as the content of a parameter (and not as a part of the SQL command).

Input Sanitisation vs SQL injection (3)

SQL injection can be used to retrieve sensitive information from databases.

```
String sqlString = "SELECT * FROM db_user WHERE username=' +  
    username + ' AND passwordhash=' + passwordhash + '";  
Statement stmt = connection.createStatement();  
ResultSet rs = stmt.executeUpdate(sqlString);
```

Suppose the user inputs validuser' OR '1'='1 as username.

```
SELECT * FROM db_user WHERE username='validuser' OR '1'='1' AND  
    passwordhash='<passwordhash>'
```

The condition will always be true because of '1'='1'.

Input Sanitisation vs XML injection (1)

- ▶ The extensible markup language (XML) is designed to help store, structure, and transfer data. You will learn about XML after the reading week.
- ▶ A user who has the ability to provide input string data that is incorporated into an XML document can inject XML tags.
- ▶ These tags are interpreted by the XML parser and may cause data to be overridden.

Input Sanitisation vs XML injection (2)

- ▶ An online store application that allows the user to specify the quantity of an item available for purchase might generate the following XML document:

```
<item>
  <description>Widget</description>
  <price>500.0</price>
  <quantity>1</quantity>
</item>
```

Input Sanitisation vs XML injection (2)

- ▶ An online store application that allows the user to specify the quantity of an item available for purchase might generate the following XML document:

```
<item>
  <description>Widget</description>
  <price>500.0</price>
  <quantity>1</quantity>
</item>
```

- ▶ An attacker might input the following string instead of a count for the quantity:

```
1</quantity><price>1.0</price><quantity>1
```


Input Sanitisation vs XML injection (2)

- ▶ An online store application that allows the user to specify the quantity of an item available for purchase might generate the following XML document:

```
<item>
  <description>Widget</description>
  <price>500.0</price>
  <quantity>1</quantity>
</item>
```

- ▶ An attacker might input the following string instead of a count for the quantity:

```
1</quantity><price>1.0</price><quantity>1
```

- ▶ In this case, the XML resolves to the following:

```
<item>
  <description>Widget</description>
  <price>500.0</price>
  <quantity>1</quantity><price>1.0</price><quantity>1</quantity>
</item>
```

Input Sanitisation vs XML injection (2)

- ▶ An online store application that allows the user to specify the quantity of an item available for purchase might generate the following XML document:

```
<item>
  <description>Widget</description>
  <price>500.0</price>
  <quantity>1</quantity>
</item>
```

- ▶ An attacker might input the following string instead of a count for the quantity:

```
1</quantity><price>1.0</price><quantity>1
```

- ▶ In this case, the XML resolves to the following:

```
<item>
  <description>Widget</description>
  <price>500.0</price>
  <quantity>1</quantity><price>1.0</price><quantity>1</quantity>
</item>
```

- ▶ An XML parser may interpret the XML in this example such that the second price field overrides the first, changing the price of the item to £1.

Input Sanitisation vs XML injection (3)

```
public class OnlineStore {
    private static void createXMLStreamBad(final
        BufferedOutputStream outputStream, final String quantity)
        throws IOException {
        String xmlString =
            "<item>\n<description>Widget</description>\n"
            + "<price>500</price>\n" + "<quantity>" + quantity
            + "</quantity></item>";
        ...
    }
}
```

```
public class OnlineStore {
    private static void createXMLStream(final
        BufferedOutputStream outputStream, final String quantity)
        throws IOException, NumberFormatException {
        // Write XML string only if quantity is an unsigned integer
        (count).
        int count = Integer.parseUnsignedInt(quantity);
        String xmlString =
            "<item>\n<description>Widget</description>\n"
            + "<price>500</price>\n" + "<quantity>" + count +
            "</quantity></item>";
        ...
    }
}
```

Data Integrity

- ▶ **Data integrity:** data is protected against damage, from corruption or hardware failure.
- ▶ Detecting errors in data:
 - ▶ Error-detecting codes
 - ▶ Cryptographic hash functions

Examples of Error-Detecting Codes

Parity bit Bit added to a group of source bits to ensure that the number of bits with value 1 in the outcome is even or odd.

Source bits	Even Parity	Odd Parity
00000000	0	1
00000001	1	0
10010111	1	0
11111111	0	1

Examples of Error-Detecting Codes

Parity bit Bit added to a group of source bits to ensure that the number of bits with value 1 in the outcome is even or odd.

Source bits	Even Parity	Odd Parity
00000000	0	1
00000001	1	0
10010111	1	0
11111111	0	1

Checksum Small piece of data calculated from block of source data.

Examples of Error-Detecting Codes

Parity bit Bit added to a group of source bits to ensure that the number of bits with value 1 in the outcome is even or odd.

Source bits	Even Parity	Odd Parity
00000000	0	1
00000001	1	0
10010111	1	0
11111111	0	1

Checksum Small piece of data calculated from block of source data.

For example, the last digit of an ISBN number is a check digit computed so that multiplying each digit by its position in the number (counting from the right) and taking the sum of these products modulo 11 is 0.

ISBN: 0-132-39077-9

$$\begin{aligned} &0 \times 10 + 1 \times 9 + 3 \times 8 + 2 \times 7 + 3 \times 6 \\ &+ 9 \times 5 + 0 \times 4 + 7 \times 3 + 7 \times 2 + 9 \times 1 = 154 \\ &\equiv 0(\text{mod}11) \end{aligned}$$

Cryptographic Hash Functions

- ▶ A **hash function** is a function used to map data of any size to data of fixed size.
- ▶ A **cryptographic hash function** is a hash function considered practically impossible to invert (and thus to recreate the original data from the hash).
- ▶ Can provide strong assurances about data integrity, whether changes of the data are accidental (e.g., due to transmission errors) or maliciously introduced.

Cryptographic Hash Functions

The screenshot shows the MSDN Subscriber Downloads page. At the top, there's a search bar with "msdn subscriptions" and a Bing logo. The navigation bar includes links like Home, My Account, Buy, Renew or Upgrade, Subscriber Downloads, My Product Keys, and Help. The main heading is "Welcome to Subscriber Downloads." Below it, a message says "Find one of the thousands of products available with your subscription by searching or browsing for a product to download." A search bar contains "Windows 10 Education (x64)" and a "Go" button. A "Browse" dropdown is set to "Products A - Z", and a "Product Categories" dropdown is also present. On the left, under "REFINE YOUR RESULTS", there are radio buttons for "All products available for subscribers" (selected) and "Products available with my subscription". Below that, under "ARCHITECTURE", there are checkboxes for "All" (selected) and "64-bit". The main content area shows "Downloads / 'Windows 10 Education (x64)' (4 results)" and "Sorted by: Release Date". The product "Windows 10 Education (x64) - DVD (English)" is listed with a file icon, language "English", release date "7/29/2015", and a "Details" link. A "Product Keys" button and a "Download" button are also visible. The file size is "3754 MB". A description states: "Windows 10 Education is available just for education customers in volume licensing programs. Windows 10 Education includes features from Windows 10 Enterprise that are ideal for advanced security, and the comprehensive device control and management needs of today's educational institution. Windows 10 Education also enables simplified deployment in the education space; this edition provides a direct path for many devices to upgrade from Windows 10 Home or Windows 10 Pro." The "File Name" is "en_windows_10_education_x64_dvd_6848120.iso". The "Languages" are "English". The "SHA1" hash is "02D3174B7853AA3465828B0AE1D3D6A0C22AFD57". "Permalinks" are provided for "File", "Download", and "Direct Download".

f(iso image) = 02D3174B7853AA3465828B0AE1D3D6A0C22AFD57

Cryptography

- ▶ 'Secret writing' is the strongest tool for controlling against many kinds of security threats.
 - ▶ Well-disguised data cannot be read, modified, or fabricated easily.
- ▶ Rooted in higher mathematics
 - ▶ Number theory, group theory, computational complexity, and even real analysis, not to mention probability and statistics.
 - ▶ Fortunately, it is not necessary to understand the underlying mathematics to be able to use cryptography.

Encryption Concepts

- ▶ **Encryption** is the process of encoding a message so that its meaning is not obvious
- ▶ **Decryption** is the reverse process, transforming an encrypted message back into its normal, original form.
- ▶ Alternatively, the terms **encode** and **decode** or **encipher** and **decipher** are used instead of encrypt and decrypt
- ▶ A system for encryption and decryption is called a **cryptosystem**.



Terminology

Steps involved in sending messages

- ▶ from a **sender**
- ▶ to a **recipient**
- ▶ If the sender entrusts the message to T, who then delivers it to the recipient, T then becomes the **transmission medium**.
- ▶ If an **outsider**, wants to access the message (to read, change, or even destroy it), we call the outsider an **interceptor** or **intruder**.

Terminology

- ▶ When a sender transmits a message via the transmission medium, the message is vulnerable to exploitation
- ▶ An outsider might try to access the message as follows:
 - Intercept:** reading or listening to it (Confidentiality)
 - Modify:** changing it (Integrity)
 - Fabricate:** authentic-looking message, arranging for it to be delivered as if it came from the sender (Integrity)
 - Interrupt:** preventing it from reaching the recipient (Availability)

Encryption can address all these problems.

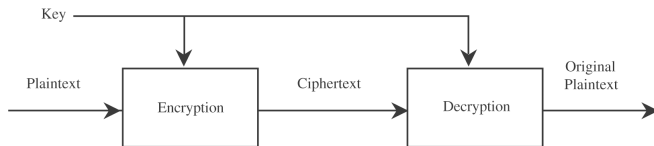
Encryption Systems

- ▶ Cryptosystem: set of rules (algorithms) for how to encrypt the plaintext and how to decrypt the ciphertext.
- ▶ Algorithms use a **key**, denoted by K , so that the resulting ciphertext depends on the original plaintext message, the algorithm, and the key value.

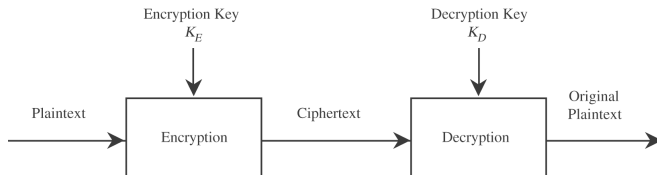
Encryption Systems: Analogy with Locks

- ▶ Expensive to contract someone to invent and make a lock just for one house.
 - ▶ Difficult to know whether a particular inventor's lock was solid or how it compared with those of other inventors.
 - ▶ **Better solution:** have a few well-known companies producing standard locks that differ according to the (physical) key.
 - ▶ People have the same model of lock, but with different keys.
- ▶ Similarly in computing security
 - ▶ useful to have a few well-examined encryption algorithms that everyone could use, but the differing keys would prevent someone from breaking into what you are trying to protect.

Two Types of Encryption Systems



(a) Symmetric Cryptosystem



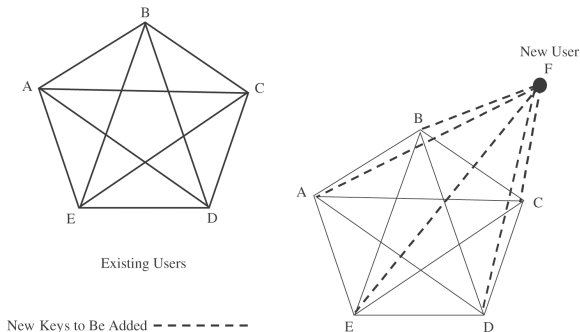
(b) Asymmetric Cryptosystem

Symmetric: the encryption and decryption keys are the same. For example, when protecting a file with a password.

Asymmetric: encryption and decryption keys come in pairs.

Symmetric Key Encryption

- ▶ In a symmetric key system, each pair of users needs a separate key.
- ▶ As the number of users grow, the number of keys grows rapidly. An n -user system has $n*(n-1)/2$ keys (key proliferation)
- ▶ It is harder to maintain security of the keys already distributed (we cannot expect users to memorise so many keys)



Asymmetric or Public Key Encryption

- ▶ Proposed in 1976, by Diffie and Hellman
- ▶ With public key, anyone using a single public key can send a secret message to a user (the message remains protected)
- ▶ Each user has two keys:
 - ▶ A public key and a private key
 - ▶ The user may publish the public key freely
 - ▶ The keys operate as **inverses**, one key undoes the encryption provided by the other

Good Practices for Password Security

[MAIN MENU](#)[MY STORIES: 24](#)[FORUMS](#)

RISK ASSESSMENT / SECURITY & HACKTIVISM

Password cracking attacks on Bitcoin wallets net £71,000

"Active attacker community" often emptied accounts minutes after they went live.

by **Dan Goodin** (US) - Feb 15, 2016 7:05pm GMT



LATEST FEATURE STORY



[FEATURE STORY \(7 PAGES\)](#)

Headshot: A visual history of first-person shooters

Doom, Halo, Goldeneye, Half-Life, Call of Duty... you may recognise a few of these.

STAY IN THE KNOW WITH



LATEST NEWS

[NEARLY HALFWAY TO 11](#)

AT&T trialling early 5G tech, promises speeds 10 to 100 times faster than LTE

[ISTARCRAFT](#)

Blizzard execs want to reinvent real-time strategy on mobile phones

Good Practices for Password Security



The electronic wallets were popularly known as “brain wallets” because, the thinking went, Bitcoin funds were stored in users’ minds through memorization of a password rather than a 64-character private key that had to be written on paper or stored digitally. For years, brain wallets were promoted as a safer and more user-friendly way to secure Bitcoins and other digital currencies although [many] Bitcoin experts had long warned that they were a bad idea.



Doom, Halo, Goldeneye, Half-Life, Call of Duty... you may recognise a few of these.

STAY IN THE KNOW WITH



LATEST NEWS

NEARLY HALFWAY TO 11

AT&T trialling early 5G tech, promises speeds 10 to 100 times faster than LTE


ISTARCRAFT

Blizzard execs want to reinvent real-time strategy on mobile phones

Good Practices for Password Security



The electronic wallets were popularly known as “brain wallets” because, the thinking went, Bitcoin funds were stored in users’ minds through memorization of a password rather than a 64-character private key that had to be written on paper or stored digitally. For years, brain wallets were promoted as a safer and more user-friendly way to secure Bitcoins and other digital currencies although [many] Bitcoin experts had long warned that they were a bad idea.



Doom, Halo, Goldeneye, Half-Life, Call of

[Researchers] showed how easy it was to attack brain wallets at scale. Brain wallets used no **cryptographic salt** and passed plaintext passwords through a single **hash** iteration (in this case, the SHA256 function), a shortcoming that made it possible for attackers to crack large numbers of brain wallet passwords at once.



ISTARCRAFT

Blizzard execs want to reinvent real-time strategy on mobile phones

Strong Passwords

- ▶ Brute force password cracking examines all possible passwords: made difficult by using long passwords using letters in lower and uppercase, numbers and symbols.
- ▶ Many users use simple passwords, common words.
- ▶ Dictionary attacks
 - ▶ Instead of brute force, try common passwords and words in the dictionary.
 - ▶ Can also take common substitutions into account, e.g., p4\$\$w0rd instead of password.
 - ▶ Mining phrases from literary works.
arstechnica.com/security/2013/10/how-the-bible-and-youtube-are-fueling-the-next-frontier-of-password-cracking

Rank	Password
1	123456
2	password
3	12345678
4	qwerty
5	12345
6	123456789
7	football
8	1234
9	1234567
10	baseball

[www.teamsid.com/
worst-passwords-2015](http://www.teamsid.com/worst-passwords-2015)

Hashing

- ▶ **Authentication:** Passwords should not be stored in plain text. Only hashed passwords should be stored (using a cryptographic hash function).
- ▶ **Encryption:** Better to not use password as key, but generate key from password.
- ▶ Given some hashing function, identical passwords produce the same hash.
- ▶ Often a user will use the same password for many different things, and many users will use the same simple passwords.

Salting

	Scenario 1		Scenario 2	
	User 1	User 2	User 1	User 2
Password	bob	bob	bob	bob
Salt	–	–	et52ed	ye5sf8
Hash	f4c31aa	f4c31aa	lvn49sa	z32i6t0

- ▶ A **salt** is random data that is used in addition to a password when creating a hash.
- ▶ Stored in plain text (accessible to attacker).
- ▶ Does not make cracking a single password harder.
- ▶ Makes cracking a list of passwords harder.
- ▶ Salt should not be too short, or reused.

Hashing (continued)

- ▶ Cryptographic hash functions often accept an iteration count as parameter.
- ▶ Value determines how slow the hash function will be.
- ▶ "Slow" hash functions slow down attackers.
- ▶ Balance between stalling attackers and having a responsive system.
- ▶ "Slow" hash functions can make Denial of Service attacks easier.

Cryptography in Java

- ▶ Java Security Architecture (`java.security`)
 - ▶ Access control, security policies, permissions, cryptographic-strength pseudo-random number generator.
 - ▶ For secure applications, use `java.security.SecureRandom`, not `java.util.Random`
- ▶ Java Cryptography Architecture (`javax.crypto`)
 - ▶ Key generation, encryption/decryption
 - ▶ Key size limited by U.S. export laws
 - ▶ Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files
 - ▶ Third-party libraries

Basic Cryptography Example in Java (1)

```
//import required packages
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;

public class Crypto {

    // Arbitrary constants
    private static int ITERS = 10000; // iteration count for
        hashing function
    private static int SALT_LENGTH = 8;

    //WARNING! Arguments to private functions are assumed to have
        been checked by the public method calling them.

    ...
}
```

Basic Cryptography Example in Java (2)

```
private static byte[] doCrypto(int opmode, char[] password,
    byte[] salt, byte[] input) throws Exception {

    // Encryption algorithm
    String algorithm = "PBEWithSHA1AndDESede";

    // Create Password-Based-Encryption Specifications from
    // password, salt and iterations
    PBEKeySpec keySpec = new PBEKeySpec(password);
    PBEPParameterSpec paramSpec = new PBEPParameterSpec(salt, ITERS);

    // Create SecretKeyFactory for PBEWithSHA1AndDESede
    SecretKeyFactory keyFactory =
        SecretKeyFactory.getInstance(algorithm);

    // Create key from keySpec with keyFactory
    SecretKey key = keyFactory.generateSecret(keySpec);

    // Create cipher and initialise it for encryption/decryption
    // according to opmode
    Cipher cipher = Cipher.getInstance(algorithm);
    cipher.init(opmode, key, paramSpec);
    return cipher.doFinal(input);
}
```

Basic Cryptography Example in Java (3)

```
private static byte[] encrypt(char[] password, byte[] input)
    throws Exception {

    // Create a random salt
    byte[] salt = new byte[SALT_LENGTH];

    // Use cryptographically strong pseudo-random number generator,
    // not the default PRNG!
    SecureRandom random = new SecureRandom();
    random.nextBytes(salt);

    byte[] ciphertext = doCrypto(Cipher.ENCRYPT_MODE, password,
        salt, input);

    // Store salt and cipher text in byte array
    ByteArrayOutputStream bOut = new ByteArrayOutputStream();
    bOut.write(salt);
    bOut.write(ciphertext);

    return bOut.toByteArray();
}
```

Basic Cryptography Example in Java (4)

```
private static byte[] decrypt(char[] password, byte[] input)
    throws Exception {

    // Read salt and cipher text
    byte[] salt = new byte[SALT_LENGTH];
    byte[] ciphertext = new byte[input.length-SALT_LENGTH];
    ByteArrayInputStream bIn = new ByteArrayInputStream(input);
    bIn.read(salt);
    bIn.read(ciphertext);

    // Decipher the cipher text
    byte[] plaintext = doCrypto(Cipher.DECRYPT_MODE, password,
        salt, ciphertext);

    return plaintext;
}
```

Going further

- ▶ Charles P. Pfleeger, Shari L. Pfleeger: Security in Computing. 4th Ed. Prentice Hall Professional, 2006
 - ▶ Available at the library as an e-book
- ▶ SEI CERT Oracle Coding Standard for Java
<https://www.securecoding.cert.org/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java>
- ▶ Honours half-module - CSCU9YS - Computer Security & Forensics (David Cairns)
- ▶ The web

Word of warning (1)

- ▶ Be careful when you use code obtained online.

Word of warning (1)

- ▶ Be careful when you use code obtained online.
- ▶ Example: you are learning to use the Unix/Linux/MacOS/PowerShell terminal and you find a webpage somewhere that has the following command to list files in a directory.

```
ls -lat
```

Word of warning (1)

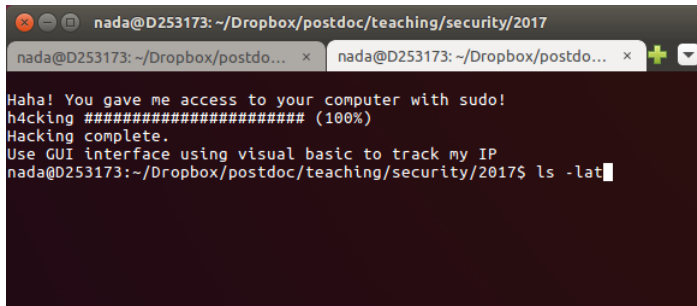
- ▶ Be careful when you use code obtained online.
- ▶ Example: you are learning to use the Unix/Linux/MacOS/PowerShell terminal and you find a webpage somewhere that has the following command to list files in a directory.

```
ls -lat
```

Word of warning (1)

- ▶ Be careful when you use code obtained online.
- ▶ Example: you are learning to use the Unix/Linux/MacOS/PowerShell terminal and you find a webpage somewhere that has the following command to list files in a directory.

```
ls -lat
```

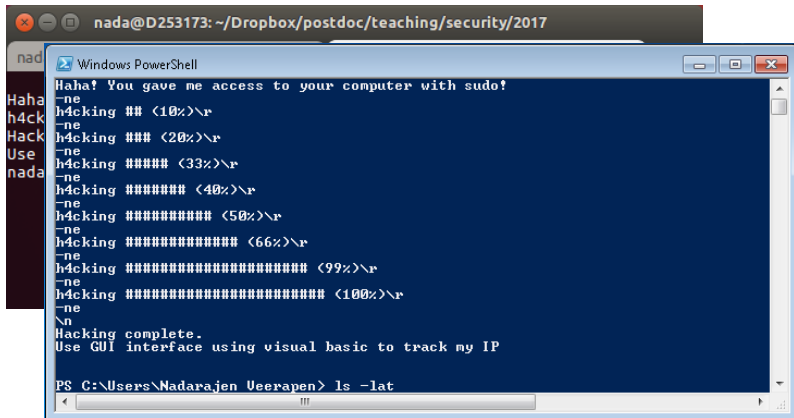
A screenshot of a terminal window with a dark background. The window title is 'nada@D253173: ~/Dropbox/postdoc/teaching/security/2017'. The terminal shows a series of messages: 'Haha! You gave me access to your computer with sudo!', 'h4cking ##### (100%)', 'Hacking complete.', and 'Use GUI interface using visual basic to track my IP'. The prompt 'nada@D253173:~/Dropbox/postdoc/teaching/security/2017\$' is followed by the command 'ls -lat' which is partially entered.

```
nada@D253173: ~/Dropbox/postdoc/teaching/security/2017
Haha! You gave me access to your computer with sudo!
h4cking ##### (100%)
Hacking complete.
Use GUI interface using visual basic to track my IP
nada@D253173:~/Dropbox/postdoc/teaching/security/2017$ ls -lat
```

Word of warning (1)

- ▶ Be careful when you use code obtained online.
- ▶ Example: you are learning to use the Unix/Linux/MacOS/PowerShell terminal and you find a webpage somewhere that has the following command to list files in a directory.

```
ls -lat
```



```
nada@D253173: ~/Dropbox/postdoc/teaching/security/2017
Haha! You gave me access to your computer with sudo!
-ne
h4cking ## (10%)\r
-ne
h4cking ### (20%)\r
-ne
h4cking ##### (33%)\r
-ne
h4cking ##### (40%)\r
-ne
h4cking ##### (50%)\r
-ne
h4cking ##### (66%)\r
-ne
h4cking ##### (99%)\r
-ne
h4cking ##### (100%)\r
-ne
\n
Hacking complete.
Use GUI interface using visual basic to track my IP

PS C:\Users\Nadarajan Veerapen> ls -lat
```

Word of warning (2)

```
<span>ls</span>
<span class="malicious">
    ; clear; echo 'Haha! You gave me access to your computer with
        sudo!';
    echo -ne 'h4cking ##' (10%)\r';
    sleep 0.3;
    echo -ne 'h4cking ###' (20%)\r';
    sleep 0.3;
    echo -ne 'h4cking ####' (33%)\r';
    sleep 0.3;
    echo -ne 'h4cking #####' (40%)\r';
    sleep 0.3;
    echo -ne 'h4cking #####' (50%)\r';
    sleep 0.3;
    echo -ne 'h4cking #####' (66%)\r';
    sleep 0.3;
    echo -ne 'h4cking #####' (99%)\r';
    sleep 0.3;
    echo -ne 'h4cking #####' (100%)\r';
    echo -ne '\n';
    echo 'Hacking complete.';
    echo 'Use GUI interface using visual basic to track my
        IP'<br> ls
</span>
<span>-lat </span>
```

Word of warning (3)

```
.malicious {  
  color: #f3f5f6; // set it to that of the page  
  font-size: 0px; // make it small  
  // move it out of the way  
  position: absolute;  
  left: -100px;  
  top: -100px;  
  height: 0px;  
  z-index: -100;  
  display: inline-block;  
  // make it un-selectable  
  -webkit-touch-callout: none;  
  -webkit-user-select: none;  
  -khtml-user-select: none;  
  -moz-user-select: none;  
  -ms-user-select: none;  
  user-select: none;  
}
```