

Introduction to cryptography

**Lessons:** 3

**Programming languages:** MakeCode, Python

**Target age:** 11-14 yrs

**Subjects & topics:**

* History: World War Two, Roman Empire
* Computational thinking: Algorithms
* Programming: Text-based programming

# Unit of work summary

In this series of three lessons aimed at students in the first year of secondary school, students learn about cryptography and undertaking practical unplugged activities to develop their logical reasoning and problem-solving skills.

They write algorithms for a Caesar cipher and are introduced to writing Caesar ciphers in text-based programming using JavaScript and Python.

This unit forms the second part of the [cyber security unit](https://microbit.org/teach/lessons/cyber-security/), though it can be used on its own and can be a good introduction to text-based programming, or a way to extend existing knowledge.

## Overall key learning

* can understand and apply the fundamental principles & concepts of computer science (logic, abstraction, algorithms, real world problem analysis and problem solving).
* practical experience of writing computer programs to solve problems.
* are responsible, competent, confident and creative users of information and communication technology.

## Additional skills

Problem-solving, collaboration, critical thinking, creative thinking, prototyping, presenting, researching

## Lesson 1: What is cryptography?

Students learn how encryption and ciphers have been used over time, especially in World War Two, before cracking and creating their own ciphers.

**Key learning:**

* To know what cryptography is and how it has been used over time to encrypt data and information
* To create and solve ciphers using logical reasoning
* To appreciate the importance of code breaking in World War Two through learning about Alan Turing

## Lesson 2: Caesar cipher algorithms

In this ‘unplugged’ lesson students create their own Caesar cipher, firstly using card before writing, testing and debugging algorithms to create a Caesar cipher to encrypt and decrypt messages.

**Key learning:**

* To create and use a paper-based Caesar cipher to encrypt and decrypt messages
* To use logical reasoning to write algorithms to encrypt and decrypt messages using a Caesar cipher
* To test and debug algorithms effectively

## Lesson 3: Ciphers and text-based programming

Students are introduced to text-based programming by exploring JavaScript code before creating a Caesar cipher in Python.

**Key learning:**

* To develop practical understanding of text-based programming
* To understand and use selection, variables and functions
* To analyse JavaScript code
* To program a Caesar cipher with Python (if you have micro:bits)

# Curriculum links

These lessons are mapped to the following learning objectives and standards:

## England National Curriculum

#### KS3 computing curriculum

Curriculum aims:

* can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
* can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
* are responsible, competent, confident and creative users of information and communication technology.

Students should be taught to:

* design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
* use logical reasoning to compare the utility of alternative algorithms for the same problem
* use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
* make appropriate use of data structures (for example, lists, tables or arrays)
* understand simple Boolean logic (for example, AND, OR and NOT) and some of its uses in circuits and programming
* understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns.

[Read the full KS3 computing curriculum](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/239067/SECONDARY_national_curriculum_-_Computing.pdf)

## Scotland Curriculum for Excellence

#### Technologies

* I can informally compare algorithms for correctness and efficiency. (TCH 4-13b)
* I understand constructs and data structures in a textual programming language (TCH 4-14a)

[Read the full Curriculum for Excellence: technologies](https://education.gov.scot/Documents/Technologies-es-os.pdf)

#### Social studies

* I can use my knowledge of a historical period to interpret the evidence and present an informed view. (SOC 3-01a)

[Read the full Curriculum for Excellence: social studies](https://education.gov.scot/Documents/social-studies-eo.pdf)

## Northern Ireland curriculum

#### Science and technology - technology and design - statutory requirements, KS3

* show deeper understanding by thinking critically and flexibly, solving problems and making informed decisions, using Mathematics and ICT where appropriate;

[Read the full technology and design statutory requirements](https://ccea.org.uk/downloads/docs/ccea-asset/General/Statutory%20Requirements%20for%20Technology%20and%20Design%20at%20Key%20Stage%203.pdf)

#### Digital skills curriculum KS3

**Become a digital maker at KS3**

* Design a digital solution for a problem using an appropriate method;

[Read the full digital skills curriculum](https://ccea.org.uk/learning-resources/digital-skills-hub/key-stage-3-digital-skills-curriculum)

#### Environment and society - history - statutory requirements, KS3

* Investigate the impact of significant events or ideas on the 20th century on the world

[Read the full history statutory requirements](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/Statutory%20Requirements%20for%20History%20at%20Key%20Stage%203.pdf)

#### Primary using ICT - desirable features - computational thinking and coding

**Level 5**

Pupils should:

* create more sophisticated coding projects using a broad range of commands and more than one platform; and
* solve a more complex problem using commands in a programming environment.

**Programmable devices (such as Parrot Drone, MicroBit or Sphero)**

* as a class look at and talk about examples of coding projects, including using multiple ‘if...then’ and ‘if...then...else’ commands, variables, sensors, events, operators and comparators;
* recognise how they can decompose these projects;
* in small groups, plan their own coding project, demonstrating a clear sense of purpose and audience, showing understanding of abstraction by deciding what details they need to include and what they can leave out, working out what different parts of the program must do and using logical reasoning to discuss and compare the commands that are required for their algorithm and predicting the outcome;
* use a range of commands to create a project, including variables, operators and control statements such as ‘if... then...’ alongside the use of ’if...then...else’ and comparators;
* test and debug at regular intervals and collaborate with others to solve problems as they arise;

**Finally**

* share their work (possibly using digital tools), respond to feedback, and comment on the work of others evaluating process and outcome; and
* organise files and publish work online (if available) so that others can view it.

[Read all Primary using ICT desirable features](https://ccea.org.uk/downloads/docs/ccea-asset/Curriculum/Primary%20Using%20ICT%20Desirable%20Features%20Update%202019.pdf)

## Curriculum for Wales

#### Science and technology

Progression step 4 - computation is the foundation for our digital world:

* I can decompose given problems and select appropriate constructs to express solutions in a variety of environments.

Progression step 5 - - computation is the foundation for our digital world:

* I can identify, define and decompose problems, choose appropriate constructs and express solutions in a variety of environments.
* I can test, evaluate and improve a solution in software.

[Read the full science and technology curriculum](https://hwb.gov.wales/curriculum-for-wales/science-and-technology/descriptions-of-learning/)

#### Digital competence framework

Progression step 4 - data and computational thinking - problem-solving and modelling:

* I can create a simple model or self-contained algorithm.
* I can identify the different parts of an algorithm to determine their purpose.
* I can detect and correct errors in algorithms.

Progression step 5 - data and computational thinking - problem-solving and modelling:

* I can develop logical solutions to determine the input, outputs and processes of a program, e.g. following pseudocode or a flowchart to come to an outcome, developing a written sequence of steps that could be followed.

[Read the digital competence framework](https://hwb.gov.wales/curriculum-for-wales/cross-curricular-skills-frameworks/digital-competence-framework)

## Code.org

Unit 1

Concepts included:

* problem solving
* inputs and outputs
* storing and processing information

[Read the full Code.org CS Discoveries curriculum](https://studio.code.org/courses/csd-2021)

## USA - CSTA Standards

#### Grades 6-8

* 2-AP-10 - Use flowcharts and/or pseudocode to address complex problems as algorithms.
* 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.

[Read the CSTA Standards in full.](https://csteachers.org/k12standards/ )

This content is published under a [Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)](https://creativecommons.org/licenses/by-sa/4.0/) licence.