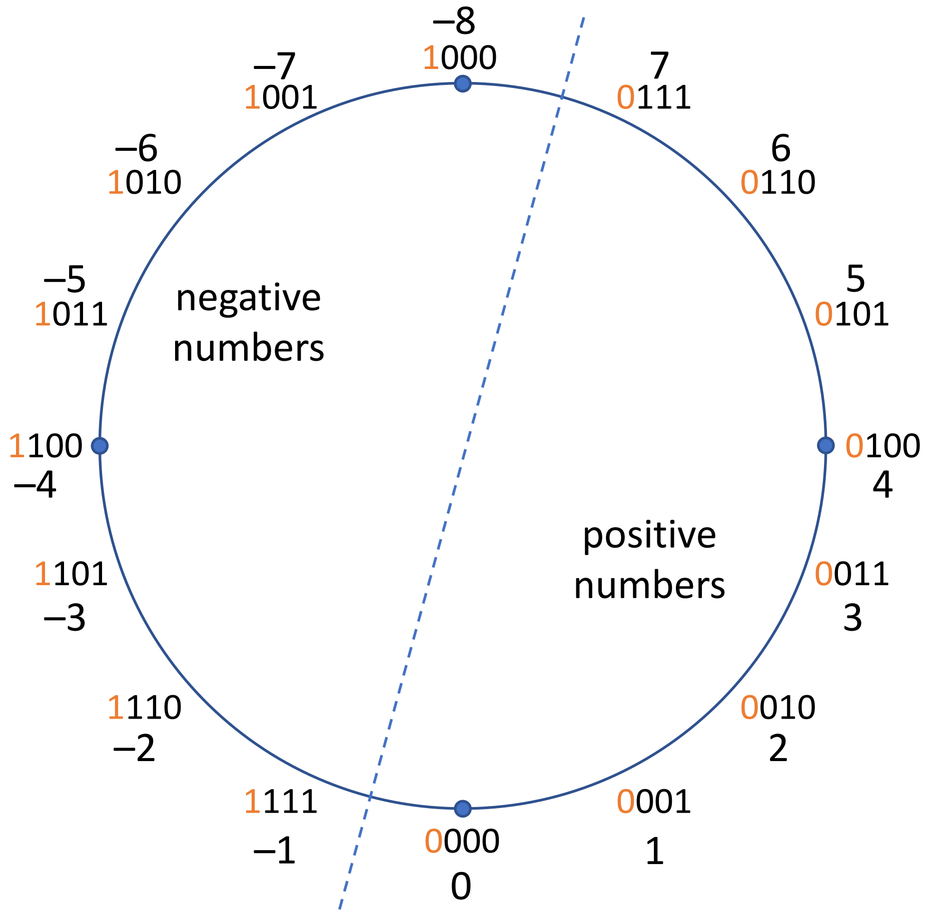
Two’s Complement: 4 Bit Example

Technische Hochschule **Rosenheim**

Two’s Complement = B-complement with base 2

0000 = 0		1000 = -8	
0001 = 1		1001 = -7	
0010 = 2		1010 = -6	
0011 = 3		1011 = -5	
0100 = 4		1100 = -4	
0101 = 5		1101 = -3	
0110 = 6		1110 = -2	
0111 = 7		1111 = -1	



leftmost Bit (= most significant bit, MSB)
0 ⟶ positive number
1 ⟶ negative number

BUT: This is not a sign-bit in the sense of sign/value-notation!

How to Obtain the Two's Complement

Technische Hochschule Rosenheim

- positive integers: conversion decimal ⟷ dual as discussed before
  - but with a fixed number of bits, i.e., leading zeros

4 Bit example

+5 = 0101

- negative integers – conversion decimal ⟶ dual
  1. convert the corresponding positive decimal to dual with fixed width
  2. apply a NOT-operation, i.e., invert all bits
  3. add one

+5 =	0101
	1010
+1	0001 =
	1011 = –5

- negative integers – conversion dual ⟶ decimal
  1. apply a NOT-operation, i.e., invert all bits
  2. add one
  3. convert to decimal and add a minus-sign

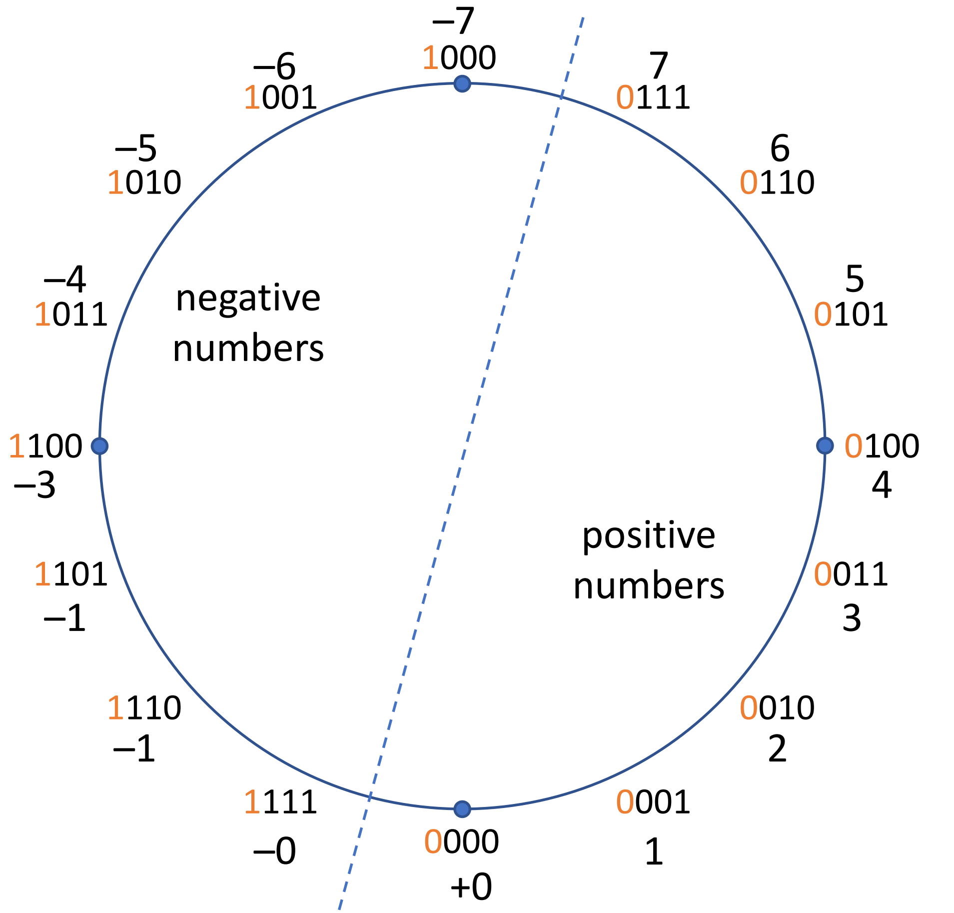
	1011 = ?
	0100
+1	0001 =
	0101 = +5
⟶	1011 = –5

Ones’ Complement: 4 Bit Example

Ones’ complement = (B–1)-complement with base 2

0000 = +0	1000 = -7
0001 = 1	1001 = -6
0010 = 2	1010 = -5
0011 = 3	1011 = -4
0100 = 4	1100 = -3
0101 = 5	1101 = -2
0110 = 6	1110 = -1
0111 = 7	1111 = -0

symmetric – modular arithmetic mod $2^N - 1$,
but we have a positive as well as a negative zero!



leftmost Bit (= most significant bit, MSB)
0 ⟶ positive number
1 ⟶ negative number

BUT: This is not a sign-bit in the sense of sign/value-notation!

How to Obtain the Ones' Complement

Technische Hochschule Rosenheim

- positive integers: conversion decimal ⟷ dual as discussed before
  - but with a fixed number of bits, i.e., leading zeros

- negative integers – conversion decimal ⟶ dual
  1. convert the corresponding positive decimal to dual with fixed width
  2. apply a NOT-operation, i.e., invert all bits

- negative integers – conversion dual ⟶ decimal
  1. apply a NOT-operation, i.e., invert all bits
  2. convert to decimal and add a minus-sign

4 Bit example

+5 = 0101

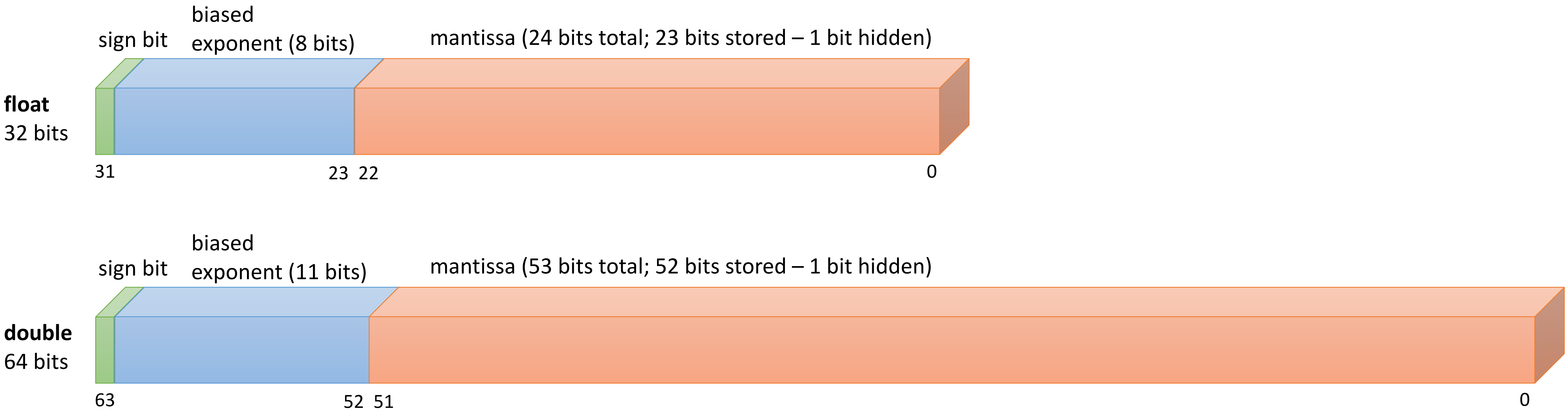
+5 = 0101
1010 = –5

1010 = ?
0101
0101 = +5
⟶ 1010 = –5

Seems much simpler than Two's complement? Well, there's a catch…

Binary IEEE Floating-point Format

- standard IEEE 754-2019
- C/C++ and Java data types use 4 bytes for float and 8 bytes for double
- the standard also defined types for half (16 Bit) and quadruple (128 Bit) precision



Fano Algorithm – Summary

1. Arrange symbols c to be encoded and the associated probabilities of occurrence $p(c)$ in a table, sorted by descending probability values.
2. Enter the subtotals of the probabilities (starting with the smallest one) in a third column.
3. Subdivide table into two parts, as close as possible to half of the respective interval.
4. Assign a 0 for all symbols above the division, and a 1 for all symbols below (or vice versa); this will form the code words from left to right.
5. Continue this procedure recursively for all partitions.
6. End when division is no longer possible.

Arithmetic Coding – Encoding Example

- Compression of the message ESSEN

c	s	U	L	
	-	1.0	0.0	… Initialization
E	1.0	0.4	0.0	
S	0.4	0.32	0.16	
S	0.16	0.288	0.224	
E	0.064	0.2496	0.224	
N	0.0256	0.2496	0.24448	

- The result is x = 0.24704

Symbol c	Probability p_i	Interval $[l(c), u(c)[$
E	$^2/_5$	$[0.0, 0.4[$
S	$^2/_5$	$[0.4, 0.8[$
N	$^1/_5$	$[0.8, 1.0[$

$$s := U - L$$
$$U := L + s \cdot u(c)$$
$$L := L + s \cdot l(c)$$

$$x := \frac{L + U}{2}$$

Arithmetic Coding – Decoding Example

Symbol c	Probability p_i	Interval $[l(c), u(c)[$
E	$^2/_5$	$[0.0, 0.4[$
S	$^2/_5$	$[0.4, 0.8[$
N	$^1/_5$	$[0.8, 1.0[$

We can gradually recover the original message from the encoded floating-point number x =0.24704)

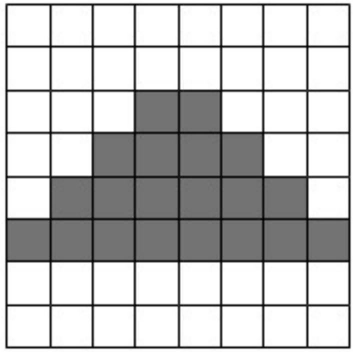
x	c (output)	$u(c)$	$l(c)$	s
0.24704	E	0.4	0.0	0.4
0.6176	S	0.8	0.4	0.4
0.544	S	0.8	0.4	0.4
0.36	E	0.4	0.0	0.4
0.9	N	1.0	0.8	0.2

Output the symbol c that corresponds to the interval where x is located

$$s = u(c) - l(c)$$
$$x \quad := \frac{x - l(c)}{s}$$

Run-length Encoding – Simple Example

- Run-length encoding of a binary image



00000000	0000
00000000	0000
00011000	001110100011
00111100	001011000010
01111110	000111100001
11111111	1000
00000000	0000
00000000	0000

- Transfer pairs of numbers (data value, run-length)
  - Run-lengths: 001=1 010=2 011=3 100=4
    101=5 110=6 111=7 000=8

- Compression from 64 to 56 bits is achieved

LZW Algorithm – Encoding Example

Encoding of the string ABABCBABAB

Symbol c	Prefix P	Output
	-	
A	A	
B	B	0
A	A	1
B	AB	
C	C	3
B	B	2
A	BA	
B	B	4
A	BA	
B	BAB	
		7

Code table

Prefix	Code
A	0
B	1
C	2
AB	3
BA	4
ABC	5
CB	6
BAB	7

Read next input character c from the input string
If Pc is already in the code table:
    Set $P := Pc$
Else:
    Insert Pc in the code table
    Output the code for P
    Set $P := c$

Output the code for the last prefix P

$\longrightarrow$ encoded string: **0,1,3,2,4,7**

LZW Algorithm – Decoding Example

Technische Hochschule Rosenheim

Decoding of 0,1,3,2,4,7

Code c	k	Prefix P	Output
		-	
0	A	A	A
1	B	B	B
3	A	AB	AB
2	C	C	C
4	B	BA	BA
7	B	BAB	BAB

Code table

Prefix	Code
A	0
B	1
C	2
AB	3
BA	4
ABC	5
CB	6
BAB	7

Read next input code word c
If c is already in the code table:
    Output the string corresponding to c
    Set k := first character of this string
    Insert Pk in the code table, is it is not yet in there
    Set P := string corresponding to code c
Else (special case):
    Set k := first character of P
    Output Pk
    Insert Pk in the code table
    Set $P := Pk$

$\longrightarrow$ decoded message: ABABCBABAB

m-out-of-n (m-oo-n) Codes

- (Nonlinear) block codes with a word length of n
- Each code word contains exactly
  - m Ones and
  - n – m Zeros
- Special case: 1-out-of-n code: „one-hot“ coding
- The code contains exactly $\binom{n}{m}$ code words

Examples:

Digit	2-oo-5 code	1-oo-10 code
0	00011	0000000001
1	00101	0000000010
2	00110	0000000100
3	01001	0000001000
4	01010	0000010000
5	01100	0000100000
6	10001	0001000000
7	10010	0010000000
8	10100	0100000000
9	11000	1000000000

2-D Parity Check – Summary

Technische Hochschule Rosenheim

- 2-D Parity Check is
  - 1-error-correcting (Correction of single errors and detection of double errors) OR
  - 3-error-detecting (Detection of single, double, and triple errors).
- Disadvantage: We must wait for whole blocks to be transferred before correction
- The concept can be generalized to more dimensions straightforwardly
  - the Hamming distance of a d-dimensional parity check is $d+1$
  - therefore, a maximum of $d/2$ erroneous bits can be corrected

Checksums in General

Check the (decimal) sequence $z_n \dots z_i \dots z_0$ (including check digits) using weights g_i

$$\sum_{i=0}^{n} g_i z_i \mod m = 0$$

- Detection of single incorrect digits is guaranteed if all weights g_i and m are relatively prime (*teilerfremd*): $\gcd(g_i, m) = 1$
- Detection of the transposition (*Vertauschung*) of two digits z_i and z_k is guaranteed, if $g_i - g_k$ and m are relatively prime.

$\longrightarrow$ using prime numbers for m makes sense

CRC – Detectable Errors

- received polynomial $S'(x) = S(x) + F(x)$
  - $F(x)$ is a polynomial that represents the erroneous bits
  - $F(x) = 0 \longrightarrow$ no errors
- all errors can be detected where $F(x)$ is not a multiple of $C(x)$
  $\longrightarrow$ Requirements for generators $C(x)$

- Which errors can be detected?
  - all single-bit errors, if x^k and the constant term 1 exist
  - all double errors, if $C(x)$ has at least three terms, and the size of the data is smaller than the cycle length of $C(x)$
  - all r-bit errors for odd r, if $C(x)$ has an even number of terms; especially if it contains the factor $(x + 1)$
  - all burst errors of length smaller k, if $C(x)$ contains the constant term
  - most burst errors of length $\geq k$

RS – Encoding

Technische Hochschule Rosenheim

- Evaluate $P(x)$ at the n positions $u_0, u_1, \dots, u_{n-1}$
  - best to use Horner's method or discrete Fourier-Transform (DFT) as Fast Fourier-Transform (FFT)
- Code word $\boldsymbol{c} = \left(P(u_0), P(u_1), \dots, P(u_{n-1})\right)$

Example: RS(q, m, n) with $q = 5,\ m = 3,\ n = 5$

- Encode message $\boldsymbol{a} = (1, 2, 3)$ $\longrightarrow$ polynomial: $P(x) = 1 + 2x + 3x^2$
- Evaluate $P(x)$ at $n = 5$ positions
  - more are not possible anyway, since the field $\mathbb{F}_5$ has only 5 elements

$$
\begin{aligned}
P(0) &= 1 + 0 + 0 && = 1 && \pmod 5\\
P(1) &= 1 + 2 + 3 = 6 && = 1 && \pmod 5\\
P(2) &= 1 + 4 + 12 = 17 && = 2 && \pmod 5\\
P(3) &= 1 + 6 + 27 = 34 && = 4 && \pmod 5\\
P(4) &= 1 + 8 + 48 = 47 && = 2 && \pmod 5
\end{aligned}
$$

- Code word $\boldsymbol{c} = (1, 1, 2, 4, 2)$

RS – Decoding – Erasure – Example

- RS(q, m, n) with $q = 5,\ m = 3,\ n = 5$ as before
- $P(x)$ was evaluated at the positions $u_i = 0, 1, 2, 3, 4$
- Sent code word was $\boldsymbol{c} = (1, 1, 2, 4, 2)$
  - the last two values were erased $\longrightarrow$ received: $(1, 1, 2, \varepsilon, \varepsilon)$

- Determine polynomials $g_i(x)$:

mod 5 !

$$
\begin{aligned}
g_0(x) &= (x-1)(x-2) = x^2 - 3x + 2 && = x^2 + 2x + 2 \\
g_1(x) &= x(x-2) = x^2 - 2x && = x^2 + 3x \\
g_2(x) &= x(x-1) = x^2 - x && = x^2 + 4x
\end{aligned}
$$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

RS – Decoding – Erasure – Example

Evaluate the $g_i(u_i)$ at $u_i = 0, 1, 2$

$$g_0(x) = x^2 + 2x + 2$$
$$g_0(0) = 2$$

$$g_1(x) = x^2 + 3x$$
$$g_1(1) = 1 + 3 = 4$$

$$g_2(x) = x^2 + 4x$$
$$g_2(2) = 4 + 8 = 12 = 2$$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

RS – Decoding – Erasure – Example

- Determinate multiplicative inverses $g_i^{-1}(u_i)$
  - they always exist because we have a field
  - use, e.g., extended Euclidean algorithm

$g_0(0) = 2 \longrightarrow g_0^{-1}(0) = 3$ (Test: 2 · 3 = 6 = 1)
$g_1(1) = 4 \longrightarrow g_1^{-1}(1) = 4$ (Test: 4 · 4 = 16 = 1)
$g_2(2) = 2 \longrightarrow g_2^{-1}(2) = 3$ (Test: 2 · 3 = 6 = 1)

- Product $P(u_i)g_i^{-1}(u_i)$

$$
\begin{aligned}
P(0)g_0^{-1}(0) &= 1 \cdot 3 = 3 \\
P(1)g_1^{-1}(1) &= 1 \cdot 4 = 4 \\
P(2)g_2^{-1}(2) &= 2 \cdot 3 = 6 = 1 \pmod 5
\end{aligned}
$$

$(1, 1, 2, \varepsilon, \varepsilon)$

$$P(x) = \sum_{i=0}^{m-1} \frac{P(u_i)}{g_i(u_i)} g_i(x)$$

RS – Decoding – Erasure – Example

Plug-in everything:

$$P(x) = \sum_{i=0}^{2} \frac{P(u_i)}{g_i(u_i)} g_i(x) = 3g_0(x) + 4g_1(x) + 1g_2(x)$$

$$\begin{aligned}
&= 3(x^2 + 2x + 2) + 4(x^2 + 3x) + (x^2 + 4x) \\
&= 8x^2 + 22x + 6 \\
&= 3x^2 + 2x + 1 \\
&= 1 + 2x + 3x^2
\end{aligned}$$

$\longrightarrow$ original message was $(1, 2, 3)$

Euclidean Algorithm for Greatest Common Divisor

- gcd(a, b) = gcd(b, a mod b)
  - for a >= b
  - Stop if b = 0
  - then a is the gcd

- Examples:
  - gcd(26, 13) = gcd(13, 0) ⟶ gcd = 13
  - gcd(26, 7) = gcd(7, 5)
    = gcd(5, 2)
    = gcd(2, 1)
    = gcd(1, 0) ⟶ gcd = 1

Review: Symmetric Encryption

- Properties
  - If you can encrypt, you can also decrypt
  - Each pair of communication partners must exchange a separate common secret key

- Assessment
  - Exchange of secret key
    - Secure channel required
    - Often, however, the channel is not secure (e.g., messenger or radio connection)

  - Key management
    - Large number of keys required
    - Problem
      - What to do if sender and recipient have not met before?
      - What if a message is to be sent to several recipients at the same time?

  - Authenticity is not guaranteed (both, sender and recipient use the same key)

- Solution: Asymmetric crypto-systems

Diffie-Hellman Key Exchange

Choose two public numbers

- a prime number p
- and an integer $g \in \{2, 3, \ldots, p-2\}$

1. Alice randomly chooses an integer $x_A \in \{2, 3, \ldots, p-2\}$ $\quad y_A = g^{x_A} \bmod p$

   x_A remains secret, y_A will be sent to Bob

2. Bob randomly chooses an integer $x_B \in \{2, 3, \ldots, p-2\}$ $\quad y_B = g^{x_B} \bmod p$

   x_B remains secret, y_B will be sent to Alice

3. Alice calculates $\quad k_{AB} = {y_B}^{x_A} \bmod p = (g^{x_B} \bmod p)^{x_A} \bmod p = g^{x_B x_A} \bmod p$
4. Bob calculates $\quad k_{AB} = {y_A}^{x_B} \bmod p = (g^{x_A} \bmod p)^{x_B} \bmod p = g^{x_A x_B} \bmod p$

The key used to exchange messages is k_{AB} (or will be derived therefrom)

Diffie-Hellman – Security

g should be a primitive root modulo p (*primitive Wurzel*)

- it must have order (*Ordnung*) $p-1$:

  $$g^{p-1} = 1 \bmod p \qquad \text{and} \qquad g^a \neq 1 \bmod p \qquad \text{for all } a < p-1$$

- i.e., g is a generator (*Generator, erzeugendes Element*)
  - repeated multiplication generates all elements of the field (*Körper*) except zero

- the total number of such elements is $\phi(p-1)$

- g is a primitive root if and only if

  $$g^{\frac{p-1}{r}} \neq 1 \bmod p$$

  for each prime factor r of $p-1$

RSA – Key Generation

1. Choose two large prime numbers p and q

2. Determine RSA modulus $n = pq$
   - n should have at least 600 (decimal) digit/2048 bits

3. Calculate Euler's function of n: $\phi(n) = (p-1)(q-1)$

4. Choose an encryption exponent c with
   - $1 < c < \phi(n)$
   - c has no common divisor with Euler's function: $\gcd\big(c, \phi(n)\big) = 1$

5. Calculate decryption exponent d as modular inverse of c wrt $\phi(n)$: $cd \bmod \phi(n) = 1$
   - e.g., using the extended Euclidean algorithm

(c, n) form the public key, d is the private key

RSA – Encryption & Decryption Example

Technische Hochschule Rosenheim

Encryption of the text `CLEO`

- Determine numerical representation: 3, 12, 5, 15
- Encrypt using public key $c = 3, n = 55$
  - `C`: $y_1 = 3^3 \bmod 55 = 27$
  - `L`: $y_2 = 12^3 \bmod 55 = 1728 \bmod 55 = 23$
  - `E`: $y_3 = 5^3 \bmod 55 = 125 \bmod 55 = 15$
  - `O`: $y_4 = 15^3 \bmod 55 = 3375 \bmod 55 = 20$
- Send 27, 23, 15, 20

Decryption of 27, 23, 15, 20 using the receiver's private key $d = 27$

$x_1 = 27^{27} \bmod 55 = 3 \longrightarrow$ `C`

$x_2 = 23^{27} \bmod 55 = 12 \longrightarrow$ `L`

$x_3 = 15^{27} \bmod 55 = 5 \longrightarrow$ `E`

$x_4 = 20^{27} \bmod 55 = 15 \longrightarrow$ `O`

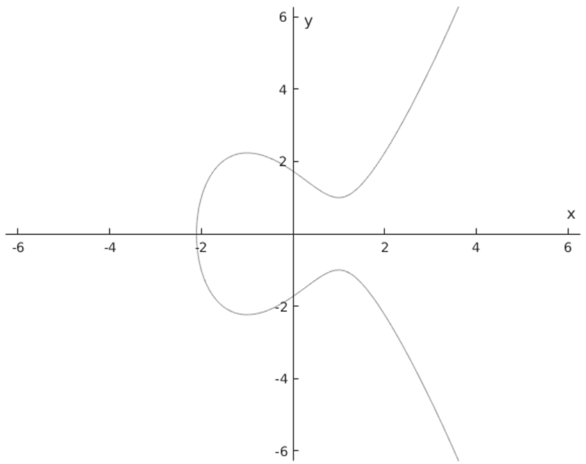
Elliptic Curve – Definition

Technische Hochschule Rosenheim

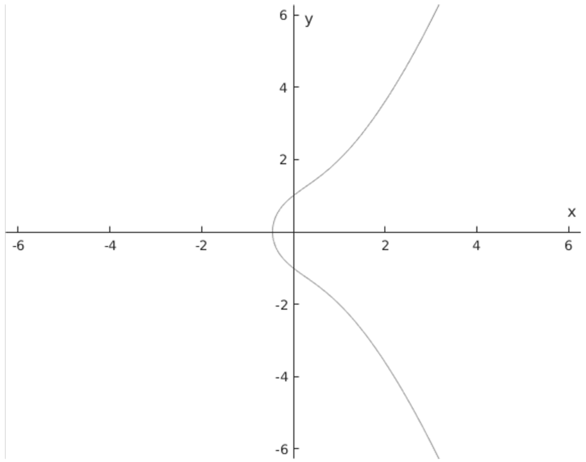
Elliptic curve ≠ ellipse!

Elliptic curve: All points (x, y) that satisfy the following equation: $y^2 = x^3 + ax + b$
with a, b, x, y elements of an arbitrary field (with at least 4 elements) and $4a^3 + 27b^2 \neq 0$

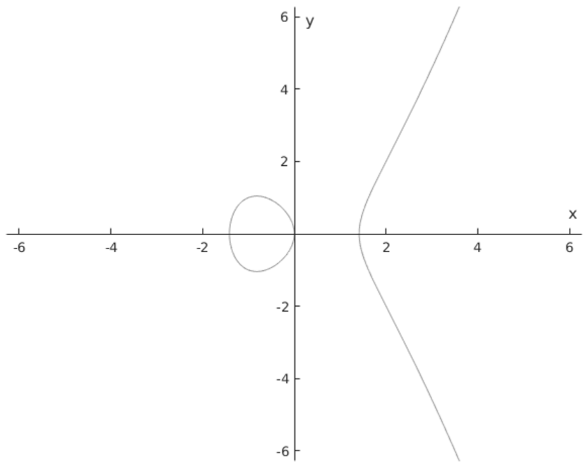
Examples (Plots over the field of real numbers):



$y^2 = x^3 - 3x + 3$



$y^2 = x^3 + 2x + 1$



$y^2 = x^3 - 2x$

Cryptography: use finite field $\mathbb{F}_q$ with $q = p^i$ elements, p prime, $i \in \{1, 2, 3, \ldots\}$ ($i = 1 \longrightarrow$ calculations modulo p)

ECC – How to Perform Calculations?

- Instead of „normal“ numbers: use points $P = (x, y)$ with $x, y \in \mathbb{F}_q$, that satisfy the equation (we'll use $q = p$ prime)
- Define a commutative group (algebraically closed, associative, neutral element, inverse)

- **Operation** „+“: $P_3 = P_1 + P_2 = (x_1, y_1) + (x_2, y_2)$ (the "+" symbol is arbitrary!), with

$$x_3 = s^2 - x_1 - x_2 \quad \mod p$$
$$y_3 = s(x_1 - x_3) - y_1 \quad \mod p$$

$$y^2 = x^3 + ax + b$$

and $s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p & \text{if } P_1 \neq P_2, x_1 \neq x_2 \text{ (point addition)} \\ \frac{3x_1^2 + a}{2y_1} \mod p & \text{if } P_1 = P_2, y_1 \neq 0 \text{ (point doubling)} \end{cases}$

- **neutral element** $\mathcal{O}$ with $P + \mathcal{O} = \mathcal{O} + P = P$
  (an infinitely distant point in the direction of the y-axis)

- **Inverse** to $P = (x, y)$ is $-P = (x, -y)$

ECC – Which Points are on the Curve?

- In $\mathbb{F}_p$ (p prime): calculations mod p!
- Insert all possible x values in $y^2 = x^3 + ax + b$
- Equation is satisfied exactly for the quadratic residues (*quadratische Reste*) R_p
  - these are numbers $c = x^3 + ax + b$ for which $c^{\frac{p-1}{2}} \bmod p = 1$ holds
  - in addition, for $c = 0$ the point $(x, 0)$ is on the curve
- For all elements of R_p: Calculate the square root (mod p!)

- Calculation of the root is easy if $4 \mid (p + 1)$
  - For $y^2 \bmod p = c$ the solutions are: $y_1 = c^{\frac{p+1}{4}}$ and $y_2 = p - y_1$

- In other cases: probabilistic algorithm, see [Wätjen 2008, Algorithmus 9.1]

- Estimate of number of elements N of the curve: $p + 1 - 2\sqrt{p} \le N \le p + 1 + 2\sqrt{p}$

  i.e., a curve consists of approx. p elements

ECC-Diffie-Hellman (ECDH)

Choose (public)

- a prime number p
- an elliptic curve $E\colon\ y^2 = x^3 + ax + b$ with N elements
- an element $g = (x_g,\ y_g) \in E$ (to be secure, it must be a primitive (= generating) element)

1. Alice randomly chooses a number $x_A \in \{2, 3, \dots, N-1\}$, adds g x_A times: $y_A = g + g + \cdots + g = x_A g$

   x_A remains secret, y_A will be sent to Bob

2. Bob randomly chooses a number $x_B \in \{2, 3, \dots, N-1\}$, adds g x_B times: $y_B = g + g + \cdots + g = x_B g$

   x_B remains secret, y_B will be sent to Alice

3. Alice calculates $k_{AB} = x_A y_B = x_A x_B g$
4. Bob calculates $k_{AB} = x_B y_A = x_B x_A g$

Since calculations are performed in a commutative group, the result is identical.
The key used to exchange messages is k_{AB} (or rather derived therefrom, e.g., from the x-value using a hash function)

ECC-Diffie-Hellman (ECDH) – Example

$p = 11, y^2 = x^3 + 3x + 9, g = (0,8)$

1. Alice randomly chooses a number $x_A \in \{2, 3, \dots, 10\} \longrightarrow 3$

$$y_A = 3 \cdot (0,8) = (0,8) + (0,8) + (0,8) = (3{,}10) + (0{,}8) = (6,10)$$

   3 remains secret, $(6,10)$ will be sent to Bob

2. Bob randomly chooses a number $x_B \in \{2, 3, \dots, 10\} \longrightarrow 2$

$$y_B = 2 \cdot (0,8) = (0,8) + (0,8) = (3,10)$$

   2 remains secret, $(3,10)$ will be sent to Alice

3. Alice calculates $\quad k_{AB} = 3 \cdot (3,10) = (2,10)$
4. Bob calculates $\quad k_{AB} = 2 \cdot (6,10) = (2,10)$

The key used to exchange messages is derived from $(2,10)$, e.g., from the x-value using a hash function

ECDH – Primitive Elements

Technische Hochschule Rosenheim

- ECDH works as presented for any public element g
- To be secure, g must be a primitive element (generator)
  - i.e., g added to itself gets zero only after all group elements have been created
  - This is the same as the primitive root criterion for standard DH
  - except that there we use multiplication and a finite field created modulo a prime (neutral element = 1), group order (= #elements , here of the multiplicative group) is $p - 1$
  - here we use point addition on the curve (neutral element = $\mathcal{O}$), group order is #points on curve + 1
- For group order N and a point g on the curve
  - determine all prime factors r of N
  - if $\frac{N}{r}g$ (= g added to itself $\frac{N}{r}$ times) is not zero ($\mathcal{O}$) for all factors r, g is primitive
  - note: $Ng = \mathcal{O}$ is always true
- In the previous example, the group contains $N = 11$ elements
  - therefore, all elements $\neq \mathcal{O}$ are primitive
  - sequence for (0, 8): (0, 8), (3, 10), (6, 10), (10, 7), (2, 1), (2, 10), (10, 4), (6, 1), (3, 1), (0, 3), $\mathcal{O}$
- It is not so easy in practice to find good curves
  - some may not have any generating elements at all

Adjacency Matrix – Directed Graphs

- Powers $\boldsymbol{A}^r$ of the adjacency matrix $\boldsymbol{A}$ give us information about existence and number of walks in directed graphs

- Number of different walks of length r from x_i to x_j = element a_{ij} of matrix $\boldsymbol{A}^r$

- Graph with n vertices is acyclic (*azyklisch*), if there exists an r with $1 \leq r < n$ such that:

  $\boldsymbol{A}^r \neq \mathbf{0}$, but $\boldsymbol{A}^s = \mathbf{0} \ \forall \ s > r$

  This is said to be a Directed Acyclic Graph (DAG)

Path Matrix

- The path matrix (*Wegematrix*) $\boldsymbol{W}$ indicates whether a path from x_i to x_j exists:

$$w_{ij} = \begin{cases} 1, \text{if there exists a path from } x_i \text{ to } x_j \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

- $\boldsymbol{W}$ can be obtained by adding up all relevant powers of the adjacency matrix
$$\boldsymbol{A} + \boldsymbol{A}^2 + \boldsymbol{A}^3 + \ldots + \boldsymbol{A}^n$$
and replacing all non-zero elements by 1