Einführung in die Programmierung

Tutorium 4

Wintersemester 2025/2026

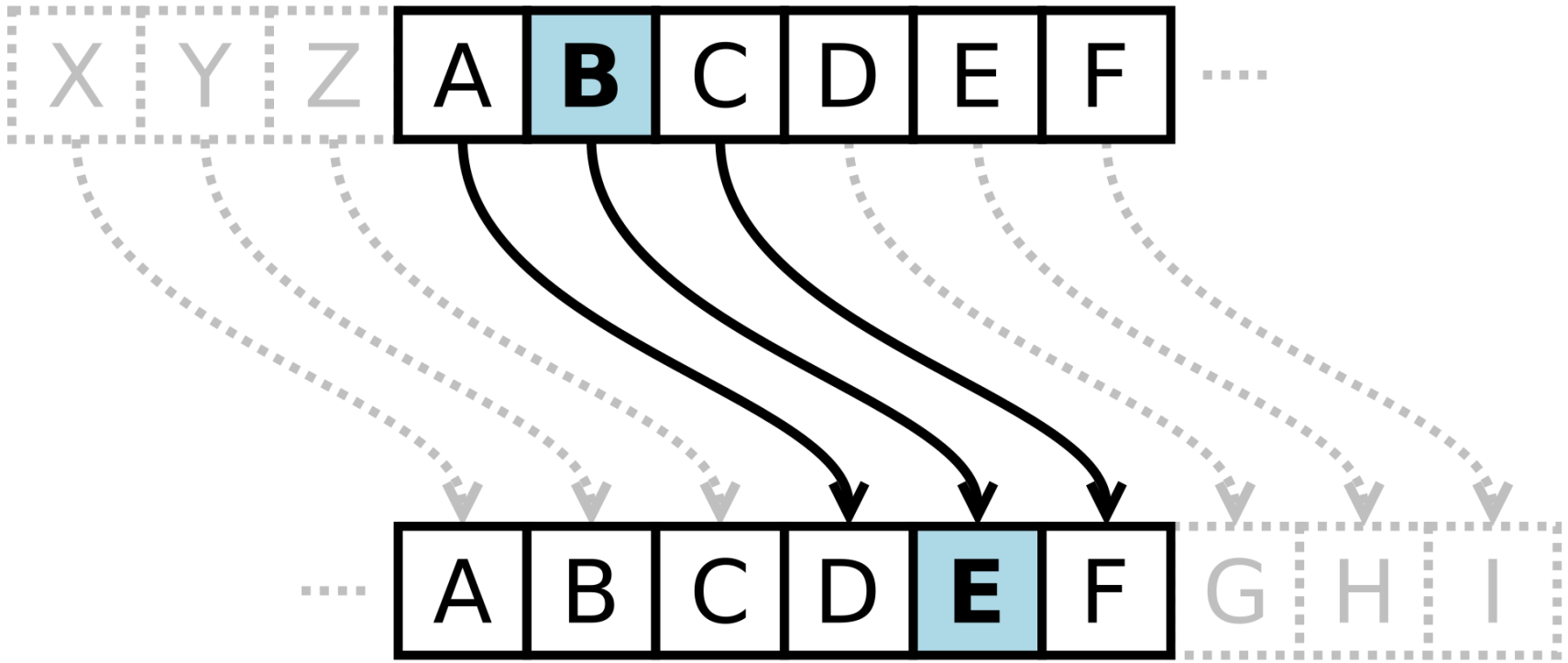
Arbeitsgruppe Systemsoftware
Angewandte Informatik 12

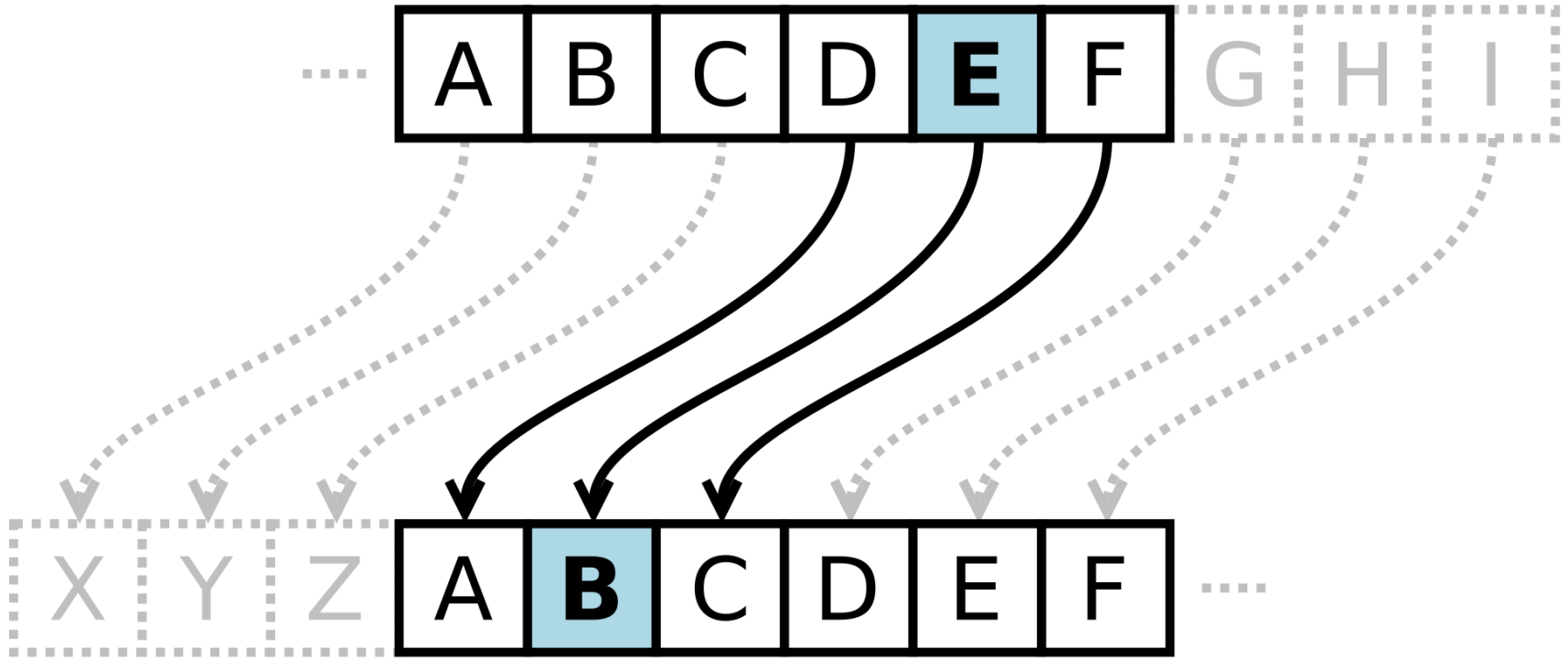
technische universität
dortmund

Arbeitsgruppe
Systemsoftware

Caesar-Verschlüsselung I

- Die Caesar-Verschlüsselung ist eine einfache Substitutionsverschlüsselung, bei der jeder Buchstabe im Alphabet um eine feste Anzahl von Positionen verschoben wird
- Julius Caesar soll diese für geheime Nachrichten verwendet haben
- Verschlüsselung und Entschlüsselung durch Verschiebung
  (z. B. bei einer Verschiebung um 3 wird A zu D)
- Leicht zu knacken, da es nur 25 mögliche Verschiebungen gibt


X Y Z A B C D E F
A B C D E F G H I


A B C D E F G H I
X Y Z A B C D E F

ASCII-Code

Hex	..0	..1	..2	..3	..4	..5	..6	..7	..8	..9	..A	..B	..C	..D	..E	..F
Hex Binär	..0000	..0001	..0010	..0011	..0100	..0101	..0110	..0111	..1000	..1001	..1010	..1011	..1100	..1101	..1110	..1111
0x0.. 0b0000..	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HAT	LF	VT	FF	CR	SO	SI
0x1.. 0b0001..	DLE	DC1	DC2	DC2	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	CS	RS	US
0x2.. 0b0010..	␣	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x3.. 0b0011..	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x4.. 0b0100..	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x5.. 0b0101..	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x6.. 0b0110..	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x7.. 0b0111..	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌫

Implementierung der Caesar-Verschlüsselung

Codeausschnitt:

```
 6 void caesarEncrypt(int shift, const char* text, char* encryptedText){
 7   for (size_t i = 0; text[i] != 0; ++i) { char base;
 8     char c = text[i];
 9     if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z')) {
10       base = (c >= 'A' && c <= 'Z') ? 'A' : 'a';
11       c = (char)((c - base + shift + 26) % 26 + base);
12     }
13     encryptedText[i] = c;
14   }
15   encryptedText[strlen(text)] = 0;
16 }
```

Text in Kleinbuchstaben/Großbuchstaben

Codeausschnitt:

```
4  void tolower(const char* text, …){
5    for (size_t i=0;text[i]!=0;i++){
6      char c = text[i];
7      if (c >= 'A' && c <= 'Z') {
8        result[i] = c – ('A' – 'a');
9      } else {
10       result[i] = c;
11     }
12   }
13   result[strlen(text)] = 0;
14 }
```

Codeausschnitt:

```
16 void toupper(const char* text, …){
17   for (size_t i=0;text[i]!=0;i++){
18     char c = text[i];
19     if (c >= 'A' && c <= 'Z') {
20       result[i] = c + ('A' – 'a');
21     } else {
22       result[i] = c;
23     }
24   }
25   result[strlen(text)] = 0;
26 }
```

Codeausschnitt:

```
41 for (int j = 0; j < n; j++) { for (int i = 0; i < m; i++) { grid[j][i] = data[k++]; }}
42 free(data);
43 for (int j = 0; j < n; j++) {
44     for (int i = 0; i < m; i++) {
45         if (i + 2 < m && grid[j][i]     == 'F'
46                       && grid[j][i + 1] == 'O'
47                       && grid[j][i + 2] == 'X') {
48             printf("\033[1;32m(%d, %d, H)\033[0m\n\r", i, j);
49             return 0;
50         }
51         if (i + 2 < m && grid[j][i]     == 'F'
52                       && grid[j + 1][i] == 'O'
53                       && grid[j + 2][i] == 'X') {
54             printf("\033[1;32m(%d, %d, V)\033[0m\n\r", i, j);
55             return 0;
56         }
57     }
58 }
```