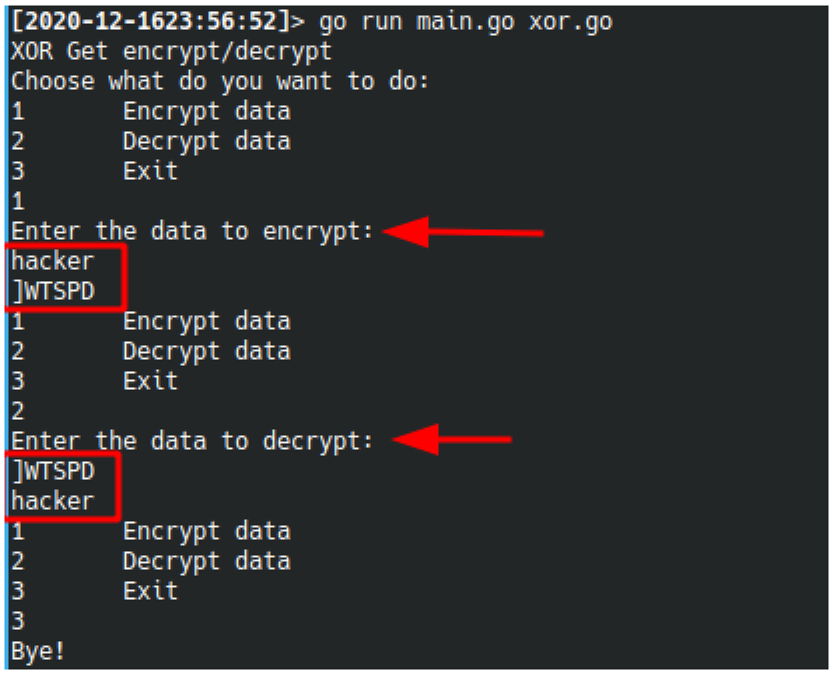
(W54A01) XOR Gate encrypt/decrypt – LAB

1. Application screnshots

The application has a menu with 3 options: encrypt, decrypt and exit. The user select the desired option and give the string to encrypt:



```
[2020-12-1623:56:52]> go run main.go xor.go
XOR Get encrypt/decrypt
Choose what do you want to do:
1       Encrypt data
2       Decrypt data
3       Exit
1
Enter the data to encrypt:
hacker
]WTSPD
1       Encrypt data
2       Decrypt data
3       Exit
2
Enter the data to decrypt:
]WTSPD
hacker
1       Encrypt data
2       Decrypt data
3       Exit
3
Bye!
```

The key to encrypt is stored in a constant in the code (xor.go), which is not a good practice, but for the sake of simplicity, it was saved in this way.

2. Source code

main.go

```
package main
import "fmt"
const menu string = `1      Encrypt data
2      Decrypt data
3      Exit`
```

```
func main() {
	fmt.Println("XOR Get encrypt/decrypt")
	fmt.Println("Choose what do you want to do:")
	var opt int
	for opt != 3 {
		fmt.Println(menu)
		fmt.Scanf("%d", &opt)
		switch opt {
		case 1:
			fmt.Println("Enter the data to encrypt:")
			var data string
			if _, e := fmt.Scanln(&data); e != nil {
				fmt.Println("There was an error with your input")
				break
			}
			fmt.Println(Encrypt(data))
		case 2:
			fmt.Println("Enter the data to decrypt:")
			var encData string
			if _, e := fmt.Scanln(&encData); e != nil {
				fmt.Println("There was an error with your input")
				break
			}
			fmt.Println(Decrypt(encData))
		}
	}
	fmt.Println("Bye!")
}
```

xor.go

```
package main

// not so secure to have it on code!
const key = "5678"

// Encrypt XOR encryption
func Encrypt(s string) string {
	var encSlice []int
	sASCII := toASCII(s)
```

```go
	keyASCII := toASCII(key)
	for i, e := range sASCII {
		encSlice = append(encSlice, e^keyASCII[i%len(keyASCII)])
	}
	return toString(encSlice)
}

// Decrypt XOR decryption
func Decrypt(s string) string {
	var decSlice []int
	sASCII := toASCII(s)
	keyASCII := toASCII(key)
	for i, e := range sASCII {
		decSlice = append(decSlice, e^keyASCII[i%len(keyASCII)])
	}
	return toString(decSlice)
}

func toASCII(s string) []int {
	var asciiVals []int
	for _, e := range s {
		asciiVals = append(asciiVals, int(e))
	}
	return asciiVals
}

func toString(is []int) string {
	var st string
	for _, s := range is {
		st += string(byte(s))
	}
	return st
}
```