MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

Criptography and Security

Laboratory work 4: Block ciphers. The DES algorithm

Elaborated:
st.gr. FAF-211 Coreţchi Mihai
Verified:
asist.univ. Cătălin Mîţu

Chișinău, 2023

The Data Encryption Standard

The Data Encryption Standard (DES) is a symmetric-key algorithm for the encryption of electronic data. Developed in the early 1970s by IBM in collaboration with the National Security Agency (NSA) of the United States, it was published as a Federal Information Processing Standard (FIPS) for the United States in 1977. DES is a block cipher—that is, a cipher that operates on plaintext blocks of a fixed size (64 bits in the case of DES) and returns ciphertext blocks of the same size. The key used for encryption and decryption in DES is ostensibly 64 bits long, though in reality only 56 bits are used; the remaining 8 bits (which are not used by the encryption algorithm) can be used for error checking or simply set to a fixed value. The core of the DES algorithm is a function that takes a 32-bit input and a 48-bit key and produces a 32-bit output. This function, which involves a combination of permutation and substitution steps, is applied 16 times in a series of operations referred to as rounds. Each round uses a different 48-bit round key generated from the initial DES key. The DES encryption process involves an initial permutation (IP) of the plaintext, followed by 16 rounds of the DES function, and a final permutation (FP) which is the inverse of the initial permutation. Decryption with DES is very similar to the encryption process, involving the same steps applied in reverse order with the round keys applied in the reverse sequence. By the late 1990s, DES was becoming less secure due to the increasing power of computational technology, which reduced the effectiveness of its 56-bit key. This vulnerability was demonstrated by the Electronic Frontier Foundation (EFF) in 1998 when they broke a DES key within 22 hours by using specialized hardware. As a result, DES is now considered to be insecure for many applications. This led to the development of Triple DES (3DES), which applies the DES cipher algorithm three times to each data block. 3DES was a recommended standard and was widely used for many years as a way to increase the security of DES without the need for a completely new algorithm. In the long term, both DES and 3DES have been replaced in many applications by the Advanced Encryption Standard (AES), which was selected by the National Institute of Standards and Technology (NIST) as the new standard for secure government communications in 2001 after a public competition. Unlike DES, AES has a block size of 128 bits and key sizes of 128, 192, or 256 bits, making it significantly more secure against brute force attacks.

Implementation

The code kicks off by setting up a multi-layered array to house the S-boxes—specialized matrices used in the DES method to jumble up the data in a way that's not straightforward to undo. These S-boxes aren't just plucked out of thin air; they're meticulously chosen to guard against certain types of cryptanalysis. Moving on, the code is equipped with a toolkit of functions that shuffle data between hexadecimal, binary, and decimal formats. In the realm of cryptography, especially in systems like DES, this kind of data juggling is crucial since various stages of the encryption demand these different numeral systems. Then there's a permutation function, a staple in the DES arsenal. This function rearranges an input sequence in a predefined order, thoroughly mixing the data up. It's this mixing—increasing the encryption's complexity—that helps keep the encrypted information out of the wrong hands. An XOR function also makes an appearance in the code. This function performs a straightforward yet powerful bitwise comparison that can be undone, making it perfect for encrypting data in a way that is reliably reversible—a must-have for any encryption algorithm. The code also describes an expansion function, mirroring a step in DES where the data's size is beefed up before being scrambled by the S-boxes. It's a bit like kneading dough with a rolling pin before cutting it into shapes—you're making sure the mix gets well combined, adding to the overall sturdiness of the encryption. When it's time for the S-boxes to do their thing, the code defines a function that emulates this crucial substitution phase. It nabs a chunk of binary string and, with a touch of magic from the S-box defined by specific rows and columns, pops out a transformed binary sequence. It's this trick—substitution—that's the very heartbeat of DES's scrambling routine. What's more, the code can whip up a random 48-bit string of zeroes and ones out of thin air. This string could be a slice of an encryption key or a piece of data waiting for the cloak of encryption. Wrapping things up, the code brings to the table an interactive element where users can get hands-on with an S-box. It invites the user to pick an S-box and then reveals how that S-box flips the input on its head. It's like a peek behind the curtain, showing off a snippet of the magic that makes DES tick.

```
/Users/mihaicoretchi/repos/CS/CS/Lab4/venv/bin/python /Users/mihaicoretchi/repos/CS/CS/Lab4/task2_9.py
Randomly generated 48-bit input (Ki + E(Ri-1)):  011100110010110010010100011011111011000110011001
Enter the S-box number (1-8): 4
The output of S-box 4 for the input block is: 1000
```

Figure 1

Result after run

Conclusion

In conclusion, the laboratory work provides a hands-on exploration of the DES algorithm, with a particular emphasis on the critical role of S-boxes. It demonstrates how these S-boxes introduce non-linearity and complexity, which are vital for the cipher's security. The practical engagement with S-boxes allows for a deeper understanding of their significance in the encryption and decryption processes. This in-depth focus on S-box operations not only clarifies their function but also showcases the ingenuity behind DES's design. Overall, the lab work reinforces the essential nature of S-boxes in cryptographic systems and the importance of understanding their role in maintaining data security.

https://github.com/eamtcPROG/CS/tree/main/Lab4