UNIVERSITY OF CALGARY

Educational Program for Cryptographic Tools Unessay

CPSC 329:
Explorations in Information Security and Privacy
–
Final Project Unessay

April 11, 2025

Authors	**Group:**	5
Clark, James	**Lecture:**	L01
Davies, Ethan		
Owen, Gwilym		
Shiells Thomas, Isaac		

Contents

1 Introduction

Cryptography secures virtually all digital communication today, yet its core concepts can seem abstract without hands-on exploration. In this project, we present a unified toolkit that brings three cryptographic techniques to life: *Frequency Analysis*, *RSA*, and the *One-Time-Pad*. We will talk about how each tool works and why they are useful.

2 Frequency Analysis

The frequency analysis tool developed in this project is designed to deconstruct any given text by counting the occurrences of each character or symbol. Essentially, the tool converts raw text into a statistical distribution, revealing hidden patterns within the data. By iterating over every character in a string and maintaining a count of its occurrences using Python's dictionary data structure, the tool generates a clear frequency profile. This profile is particularly valuable in the context of classic substitution ciphers, as certain letters—such as *'E'* in English—tend to appear more frequently than others. In a ciphertext, such frequency patterns can guide cryptanalysts in making informed guesses about the correspondence between the encrypted characters and common letters in the target language, thereby aiding the cryptographic attack process.

In any natural language, some letters and symbols occur more frequently than others. For instance, English texts typically feature *'E'* as the most common letter, followed by letters like *'T'*, *'A'*, and *'O'*. A transposition cipher which scrambles all the letters in a plaintext will perfectly preserve these natural frequencies. This means that even though the letters have been replaced, the relative proportions of each character remain fully intact. By carefully counting how many times each symbol appears in the ciphertext, it is possible to generate a frequency profile—a kind of statistical fingerprint of the text. This profile can then be compared to the known frequency distribution of the target language. This makes it easier to decipher the original ciphertext.

3 One-Time Pad

This OTP tool, implemented in Python, provides two complementary interfaces — *textPad*

for alphanumeric messages and *digitalPad* for binary/hexadecimal data — anchored by the *OTP-main* menu. In text mode, the program sanitizes user input to include only alphanumeric characters, then either accepts a user-provided pad or generates a truly random pad of equal length (repeating it if necessary). Encryption maps each character to its index in lettercodes, adds the corresponding pad index modulo 36, and converts the result back to a character; decryption subtracts the pad index instead of adding it. In digital mode, the tool auto-detects whether the input is binary, octal, decimal, or hexadecimal, parses it into an integer, and similarly accepts or generates a random key of matching bit-length. It then performs a bitwise XOR between the message integer and the key, outputting ciphertext and pad in both binary and hexadecimal formats. Both modes loop to allow repeated operations until the user chooses to exit—achieving perfect secrecy across both text and digital data.

The one-time pad achieves perfect secrecy by combining plaintext with a truly random, single-use key at least as long as the message, ensuring that the resulting ciphertext is statistically independent of — and thus reveals no information about — the original text. By providing both a modular-arithmetic interface for alphanumeric messages and a bitwise-XOR interface for binary/hex data, this tool vividly demonstrates the three critical requirements for OTP security—genuinely random key generation, strict one-time use, and secure key management—offering a hands-on exploration of information-theoretic security and one-time-pad remains the only cryptosystem to have perfect secrecy.

4 RSA

This RSA tool, implemented in Zig, parses and formats both public and private keys based on industry standards (RFC 4253 for public keys and RFC 8017 for private keys). It converts text messages into big integers and then uses modular exponentiation to perform encryption and decryption. Key functions include decoding base-64 PEM formatted keys, extracting components like exponents and moduli, and efficiently computing large-number arithmetic (such as computing multiplicative inverses via the extended Euclidean algorithm), all of which work together to securely transform data.

RSA is very valuable for cryptography because it facilitates secure communication through the use of asymmetric key pairs — one key for encryption and a different one for decryption. The method relies on complex mathematical operations, particularly the difficulty of factoring large prime numbers, making it practically infeasible to reverse without the corresponding private key. This inherent challenge ensures that only those with the proper private key can decrypt and access the data. These properties make RSA great for protecting data transmissions, verifying digital signatures, exchanging keys for other non-public-key cryptosystems, and supporting secure authentication protocols across various applications.

5 Resources

5.1 Frequency Analysis

- *Frequency Analysis* [1]

5.2 One-Time-Pad

- *One-Time Password vs. One-Time Pad: What's the Difference?* [2]

5.3 RSA

- *RFC 4253: The Secure Shell (SSH) Transport Layer Protocol*[3]

- *RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2* [4]
- *ASN.1 — Wikipedia, The Free Encyclopedia* [5]
- *X.690 — Wikipedia, The Free Encyclopedia* [6]
- *PKCS 1 — Wikipedia, The Free Encyclopedia* [7]
- *RSA (cryptosystem) — Wikipedia, The Free Encyclopedia* [8]
- *RSA Encryption* [9]

6 Contributions

6.1 Clark, James

- Frequency Analysis
- GUI/App

6.2 Davies, Ethan

- Writing Portion

6.3 Owen, Gwilym

- One-Time-Pad

6.4 Shiells Thomas, Isaac

- RSA
- One-Time-Pad (Refactor)
- Code Cleanup
- Latex
- File Organization

7 Resources

7.1 RSA

[3] was used for something

[1]

References

[1] Administrator. "Frequency analysis," 101 Computing. (Nov. 2019), [Online]. Available: https://www.101computing.net/frequency-analysis/.

[2] Rublon Authors. "One-time password vs. one-time pad: What's the difference?" Rublon. (Jan. 2024), [Online]. Available: https://rublon.com/blog/one-time-password-vs-one-time-pad-whats-the-difference/.

[3] C. M. Lonvick and T. Ylonen, *The Secure Shell (SSH) Transport Layer Protocol*, RFC 4253, Jan. 2006. DOI: 10.17487/RFC4253. [Online]. Available: https://www.rfc-editor.org/info/rfc4253.

[4] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, *PKCS #1: RSA Cryptography Specifications Version 2.2*, RFC 8017, Nov. 2016. DOI: 10.17487/RFC8017. [Online]. Available: https://www.rfc-editor.org/info/rfc8017.

[5] Wikipedia, *ASN.1 — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=ASN.1&oldid=1265445360, [Online; accessed 20-March-2025], 2025.

[6] Wikipedia, *X.690 — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=X.690&oldid=1245595955, [Online; accessed 20-March-2025], 2025.

[7] Wikipedia, *PKCS 1 — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=PKCS%201&oldid=1279978614, [Online; accessed 20-March-2025], 2025.

[8] Wikipedia, *RSA (cryptosystem) — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=RSA%20(cryptosystem)&oldid=1281118546, [Online; accessed 20-March-2025], 2025.

[9] Brilliant.org, *RSA Encryption*, Accessed: 2025-03-20. [Online]. Available: https://brilliant.org/wiki/rsa-encryption/.