"RSA... MADE SIMPLE": AN ONLINE EDUCATIONAL AND TRAINING APPLICATION

D. Sarioglou[1], E. Papaioannou[2], N. Karanikolas[1], C. Kaklamanis[2]

[1]University of Patras (GREECE)
[2]University of Patras and CTI "Diophantus" (GREECE)

Abstract

Cryptography, a key element of secure digital communication and modern online economies, relies on techniques like the RSA algorithm to ensure privacy and trust in an increasingly connected world. In 1978, Ron Rivest, Adi Shamir, and Leonard Adleman introduced their groundbreaking public-key cryptosystem, which bears their initials. The RSA algorithm nicely couples the easiness of exponentiation with the hardness of factorization of large prime numbers to guarantee that communication of sensitive information that happens in public - i.e., over internet - cannot be intercepted or disrupted in practical, real-world settings given the technology available so far.

Due to its importance regarding the security of modern internet-based communication infrastructure, the RSA algorithm is a basic section in Cryptography courses in higher education. However, the required background in number theory and the involved notation usually makes it hard for students to grasp the underlying idea and apply RSA in the context of relevant problem solving.

Motivated by these observations and with an intention to make the RSA algorithm simple for students to understand and use, we designed and implemented an online application which enables students to familiarize with concepts and the application of the RSA algorithm. Our application includes educational, training and evaluation sections presented within an elegant interface enriched with visualization features. Within the educational section, we present a comprehensive description of the RSA algorithm together with indicative examples. The training section provides the application framework and focuses on letting users familiarize with the computational blocks of the RSA algorithm; the training section can be also used as an online simulator helping students crosscheck their solutions to course exercises and problems. The evaluation section contains a test enabling students to verify their understanding and knowledge regarding the overall RSA framework. Furthermore, a calculator functionality is provided for exponentiation and modulo operations. Compared to existing relevant works, the novelty of our application mainly lies in its graphical interface and game-like nature. Extending the concept of existing approaches, like for example “RSA visual and more” and “Learn RSA Cryptography”, which focus on providing results, our application places special emphasis on assisting users to comprehend the essence of the algorithm by interacting with the whole computational procedure taking place. Graphics and visualizations aim to capture the attention of users and thus obtain their engagement in completing all available sections.

Our application is responsive and thus can be used on all user devices (smartphones, tablets, and computers) regardless of their screen size. The application has been developed using the React free and open-source front-end JavaScript library as the main development environment and Github Pages for online hosting.

Keywords: Cryptography courses, RSA public-key cryptosystem, educational and training web app, React.

1 INTRODUCTION

This paper focuses on the design and implementation of a web application with a graphical interface to demonstrate the functionality of the RSA encryption algorithm. The RSA algorithm is one of the most widely used encryption methods in modern cryptography. Named after its creators, Ron Rivest, Adi Shamir, and Leonard Adleman [3], RSA operates on the principle of using two keys: a public key for encryption and a private key for decryption. Initially developed to address the limitations of symmetric encryption, where the same key is used for both encryption and decryption, RSA provided a more secure alternative by ensuring that only the intended recipient, with the corresponding private key, could decrypt messages. Since its introduction in 1977 and publication in 1978, RSA has been fundamental to online security, particularly with the rise of the internet in the 1990s [4]. It is

extensively applied in various domains, including e-commerce, banking transactions and digital signature generation. The core of RSA lies on the fact that while exponentiation of large numbers is computationally easy, factoring large prime numbers remains a computationally hard (i.e., time consuming) task. Although factorization is theoretically solvable, the time required to factor sufficiently large numbers renders such efforts impractical for potential attackers.

Due to its foundational role in secure communication systems, RSA is a central topic in university cryptography courses. Despite its conceptual simplicity and elegance, students often find it difficult to grasp and apply the RSA algorithm. This difficulty may arise due to unfamiliarity with modular arithmetic (i.e., modulo operation) or due to focusing on mathematical formulas without a previous clear understanding of the algorithm's underlying philosophy.

As part of our research, we spotted several related applications, among which CryptTool Online and Learn RSA Cryptography are noteworthy examples. CryptTool Online [6] is a web-based application that showcases various encryption algorithms, including RSA. It enables users to perform encryption and decryption operations using these algorithms and generate cryptographic keys, as well. Learn RSA Cryptography [14] is a mobile application with gamification elements that focuses specifically on the RSA algorithm. It enables users to encrypt phrases and generate keys, offering an interactive approach to understanding RSA encryption. However, none of these applications are educational in nature. Our group has already presented applications with a similar educational philosophy addressing topics from courses such as theory of computation and parallel algorithms. For instance, the N2D application [2] facilitates the transformation of non-deterministic finite automata (NFA) into their deterministic counterparts (DFA), while the FaRe application [5] enables the conversion of finite automata into equivalent regular expressions (RE). Both tools allow users to input a custom-defined NFA and return a visual representation of either the corresponding DFA or the equivalent regular expression, respectively. These applications further support step-by-step visualization of the transformation process, enhancing students' understanding and enabling experimentation or verification of their own solutions, particularly in the context of courses relevant to Theory of Computation. In addition, the FaRe application specifically focuses on the conversion of non-deterministic finite automata to equivalent regular expressions, thereby providing deeper insight into the expressive power of automata. Another relevant application, Sorting Easy [1], supports experimentation with two parallel sorting algorithms for 2D meshes, enabling users to visually inspect the operations performed by these algorithms.

RSA not only plays a pivotal role in securing data in an era of rapid digital information exchange, but it also represents a fundamental concept in the academic study of cybersecurity. Given the challenges students face in mastering RSA, there is a need for resources that bridge the gap between theory and practice. To address this need, we propose an interactive educational application which enables students to engage with RSA more intuitively, reinforcing both theoretical knowledge and practical skills. Ultimately, our goal is to facilitate a deeper understanding the role of RSA in modern cryptography and secure digital communication.

The rest of the paper is structured as follows. In Section 2, we briefly describe how the RSA algorithm works. In Section 3, we present an overview of the application, followed by a detailed description of its functionalities in Section 4. In Section 5, we outline the development process, while we conclude in Section 6 also discussing potential directions for future work.

2 HOW RSA WORKS

RSA is an asymmetric encryption algorithm commonly used for securing data and ensuring the confidentiality of communications. The algorithm relies on the mathematical properties of prime numbers and the complexity of factoring large composite numbers, making it highly resistant to attacks. By employing a system of public and private keys, RSA is utilized not only for encrypting messages but also for implementing digital signatures, which provide data authentication and integrity. It is a cornerstone of modern cryptography and is applied in various domains, including HTTPS and secure email communications.

RSA operates with a pair of a public and a private key. The public key is used for encryption, while the private key is used for decryption and remains known only to the recipient. For example (Fig. 1, left) if user A wants to send a secure message to user B, A encrypts the message using B's public key. Then, only B can decrypt the message using its own private key. This ensures that only the intended recipient can access the original content of the message.

The algorithm follows three main stages: key generation, encryption, and decryption (Fig.1, right). During key generation, both the public and private keys required for encryption and decryption are created. The encryption phase involves the sender encrypting data with the recipient's public key, resulting in a ciphertext. During decryption, the recipient uses their private key to transform, i.e., decrypt, the ciphertext back into the original message.

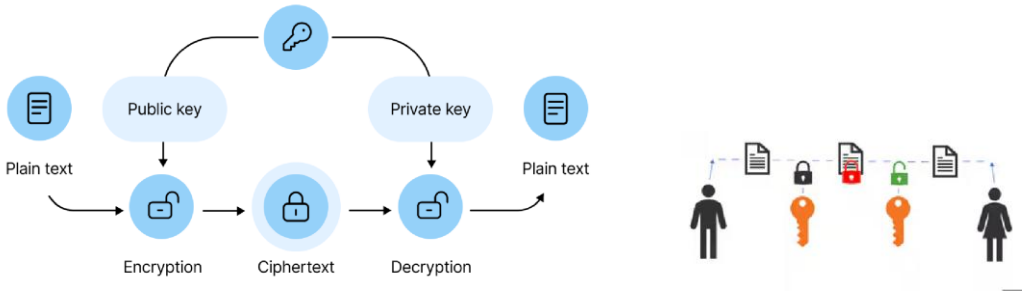

Figure 1. How RSA works.

Next, we describe how RSA works via an example illustrated in Fig. 2. For computing the public and private keys, we first choose two (in practice, large) prime numbers (i.e., numbers that have only two factors, that is, 1 and the number itself), say p = 3 and q = 11, and compute n = p * q = 3 * 11 = 33 and φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20 (where φ(n) is the Euler's totient function). Then, we choose E such that 1 < e < φ(n) and E and φ (n) are co-prime (i.e., sharing no common factors other than 1). Let E = 7. Then, we compute a value for D such that (D * e) mod φ(n) = 1 (note that by x mod y we denote the remainder of the division of x by y). One such value is D = 3 since (3 * 7) mod 20 = 1. We now have the public key denoted by (E, n) = (7, 33) and the private key denoted by (D, n) = (3, 33). The encryption of a message M, say M=13, is computed as M^E mod n resulting in the ciphertext C = 13^7 mod 33 = 7, while the decryption of C is computed as C^D mod n returning the original message m = 7^3 mod 33 = 13.

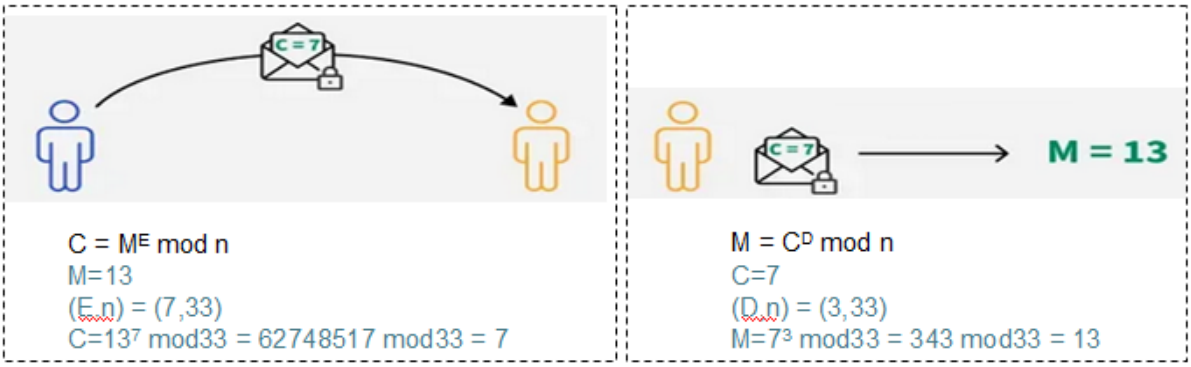

Figure 2. Message encryption and decryption using RSA.

3 THE APPLICATION

"RSA… made simple" is a web application available at https://despina-sar.github.io/RSA with a simple graphical interface and educational character. Our application, whose home screen on various devices is depicted in Fig. 3, is currently available in english and greek. It is responsive and supports accessibility features.

The interface of the application requires interaction with visitors and contains animation elements to attract interest and attention. The philosophy of the application follows the approach "description-example-application", in the sense that we first describe how the algorithm works also providing examples of its operation involving calculations and then ask for its application. In addition there is a calculator specifically designed for exponentiation and modulo operations extensively used in the RSA algorithm, as well as a special evaluation section.

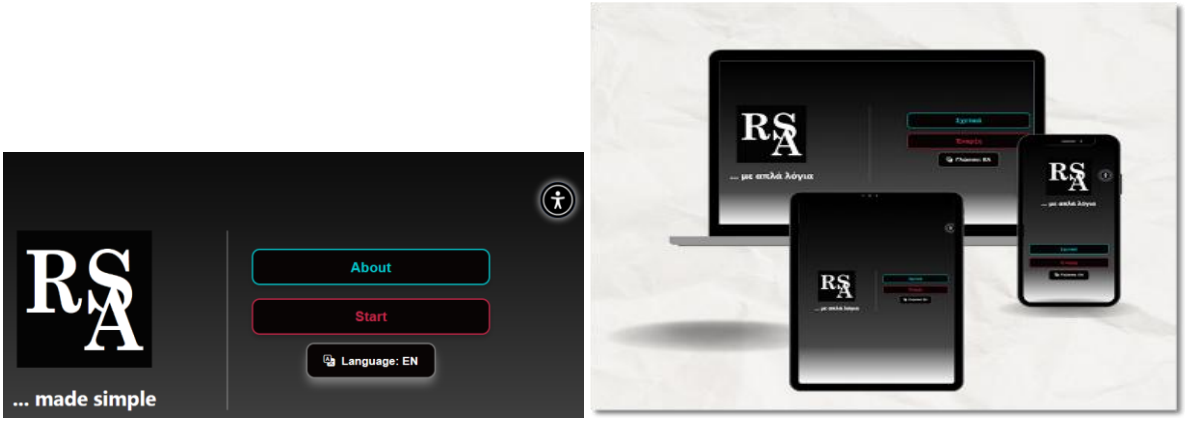

Figure 3. "RSA... made simple" home screen on various devices.

3.1 Design, user interface and functionalities

The home screen of the "RSA… made simple" application contains its logo, pointer to the accessibility menu, language selector and two main buttons labeled "About" and "Start". The green "About" button points to an introductory section which serves as a guide, featuring an infographic and a short description of each of the three stages of RSA together with corresponding examples (Fig. 4).

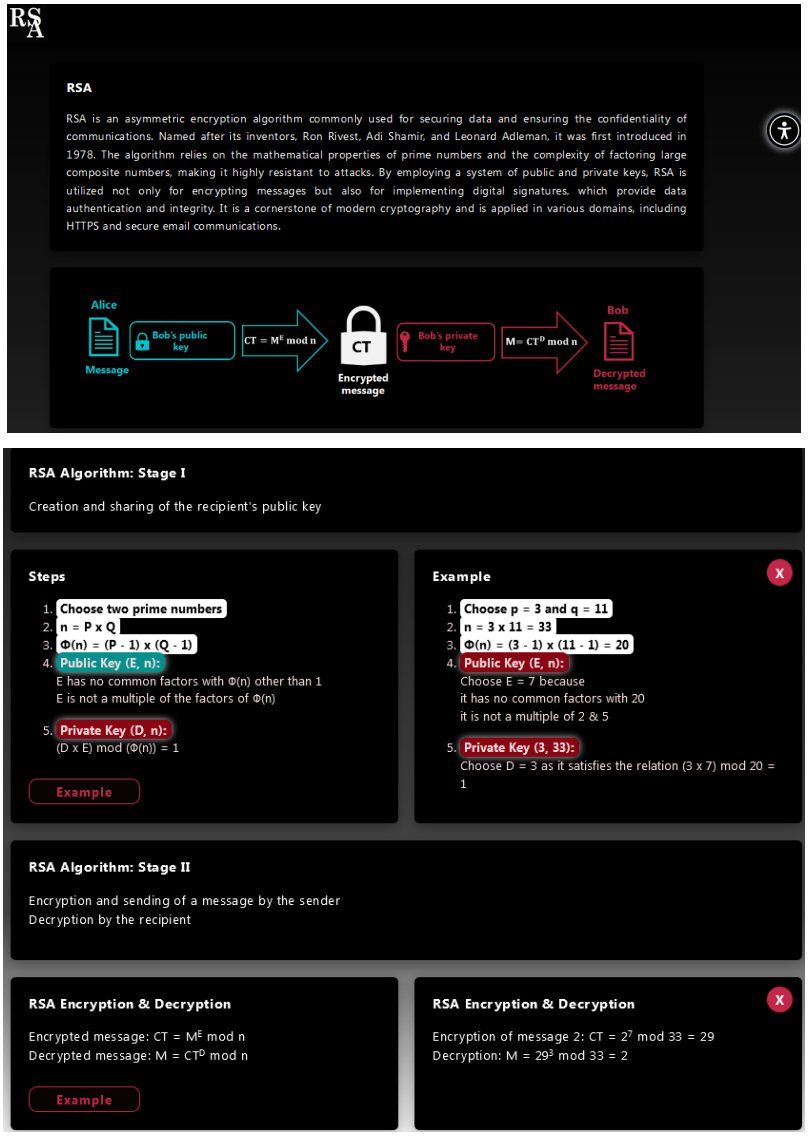

Figure 4. "RSA... made simple": About.

The red “Start” button points to the main section of the application (Fig. 5) where users can experiment with the RSA algorithm, via the standard users Bob and Alice, borrowed from the relevant literature.

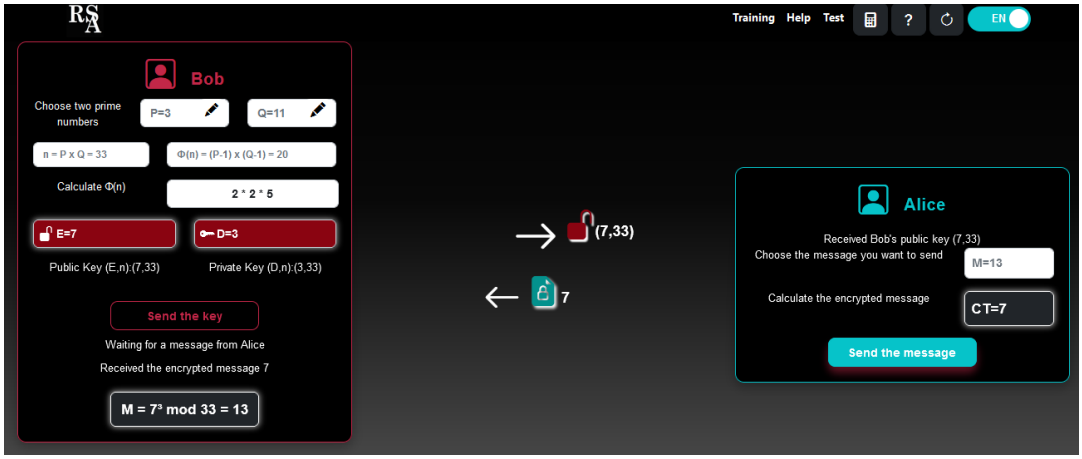

Figure 4. “RSA... made simple”: main “Training” section.

A navigation bar (Fig. 5), which is positioned at the top right of the screen and remains visible on all screens of the application, contains three items, namely “Training”, “Help”, and “Test”. It also contains three additional icons, one for the calculator, a question mark icon for the RSA infographic and an icon for refresh functionality, as well as the language selector.

Figure 5. Navigation bar.

3.1.1 Training

The “Training” section is the main section shown in Fig. 4. It is based on the classic example where two users Bob and Alice want to exchange messages with each other securely. For Alice to send messages to Bob she must have his public key. Then Bob must use his private key in order to read the message received from Alice.

The card on the left of Fig. 6 shows all operations Bob has to perform according to the RSA algorithm for calculating his public and private key. The private key remains secret while the public key is sent to Alice, who uses it to encrypt her messages to Bob. Bob then decrypts them with his own private key. All these processes i.e., public and private key computation as well as encryption and decryption of messages, are performed in this section while users using our application for training and practice.

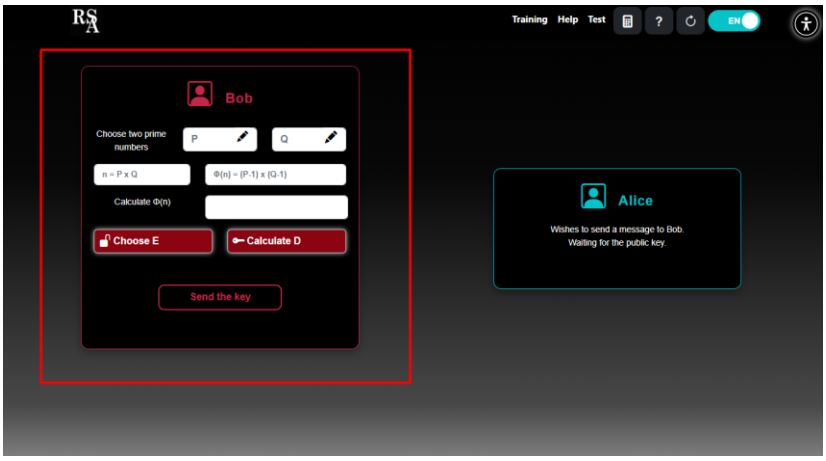

Figure 6. Bob's card on «Training» section.

More precisely, users fill in the fields according to the instructions provided, including two prime numbers, P and Q, and select Bob's public key to compute his private key. The prime numbers range from 0 to 100 for ease of calculation. If users prefer not to enter the numbers manually, they can click the pencil icon for automatic completion. Then n and Φ(n) are automatically calculated by the

application. If an invalid number is entered, an error message prompts users to select the correct value. When all fields are completed, users click “Send the key”. If any field is empty or incorrect, an error prevents further progress. Alice's card then appears. Alice receives Bob's public key and is prompted to encrypt her message. Users must type-in the message and encrypt it using Bob's public key. The message must be a positive integer not greater than 100. Error messages are displayed for invalid entries. Once Alice's card is correctly populated, users click “Send Message” to send the encrypted message to Bob. In this way an RSA encryption-decryption cycle is completed. User can click the “Refresh” icon from the navigation bar to restart the process with new data.

3.1.2 Help

In the “Help” section, all steps that users are asked to perform in the “Training” section can be performed exclusively by the application. That is, the keys are automatically calculated and the encryption and decryption process is performed automatically. There is no time limit on each step. The process proceeds step by step by users via the next button (Fig. 7).

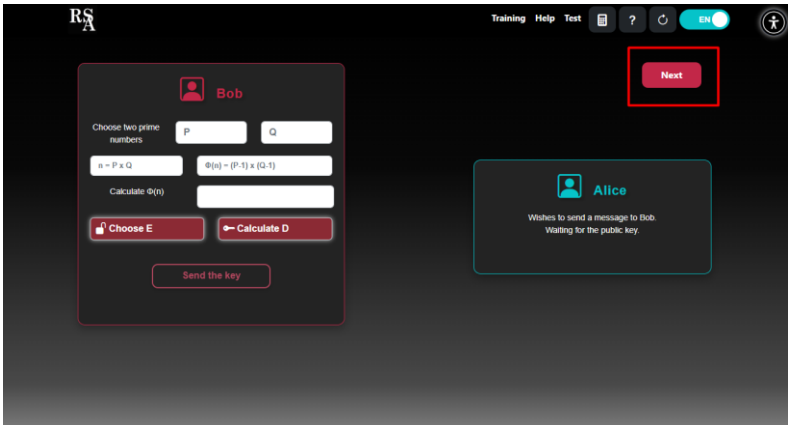

Figure 7. “RSA... made simple”: “Help” section.

3.1.3 Test

The ”Test” section contains a short test of nine questions through which users can test their knowledge of the RSA algorithm. These questions either require users to type in the answer (Fig. 8, left) or are "true-false" questions (Fig. 8, right). Although the basic philosophy of the questions remains the same, i.e. key computation, encryption or decryption of messages, on each re-execution of the test the data values are different. For example, some questions may give the keys and ask to encrypt and decrypt a message, while other questions may ask for key computation. It may also be that values for the keys are given and users are asked to tell whether these values are correct or incorrect, or that an unencrypted or decrypted message is given and users are asked to decide whether it is correct or incorrect. Questions can be answered in any order, with no time limit. Users can navigate between questions as many times as they like until they submit their answers for evaluation.

Figure 8. “RSA... made simple”: “Test” section, fill-in and “true-false” questions.

In “type-in” questions users must correctly calculate the required value and fill in the field. The “Check” button allows users to check their answer before final submission. If the answer is correct, the box turns green and locks. If the answer is wrong, an error message appears with a hint button, enabling users to view the correct answer if they wish. In “true-false” questions, where users decide if the displayed RSA-related statements are correct or not, an appropriate message appears depending on

the answer. Once all questions are answered, users move to the final card and press the “Submit” button. If not all questions are answered, a message prompts users to complete them. This informs users of the remaining unanswered questions and gives them the option to submit their answers or return to the test to finish. Once all answers are submitted, the user-score is displayed. Users can either review their answers or restart the test.

Our goal in designing the “Test” section was to create nine questions that would help users practice and assess their knowledge of the RSA algorithm while keeping them engaged. The first five questions require users to enter the correct answer, while the remaining four are true/false questions. To avoid repetition and maintain interest, we implemented a feature that generates different data each time users access the test. Specifically, nine sets of RSA parameter values are dynamically generated, including p, q, n, Φ(n), public and private keys, and a random message with its encrypted counterpart. The prime numbers p and q are restricted to [2, 3, 5, 7, 11, 13, 17, 19], and the message is a random number up to 20. This selection ensures that the generated data is easy for users to evaluate since large numbers are avoided. A subset of these values is displayed for each question.

For the questions that require input, the application compares the user's answer to the correct value and displays the corresponding message. For example, if we generate the following values: p = 17, q = 7, n = 119, Φ(n) = 96, e = 5, d = 77, M = 9, and CT = 25, and ask users to compute the private key d given n = 119, Φ(n) = 96, and e = 5, the answer will be checked against d = 77. If correct, a confirmation message is displayed. This process applies to all questions that require user input, except the first question. For the first question, users are given p and q and must compute the public key. Since the answer is not unique, we've created a function that checks whether the user's answer for e is a positive integer that satisfies two conditions: $1 < e < \Phi(n)$ and $\gcd(e, \Phi(n)) = 1$.

For true/false questions, we wanted to make sure that the answers don't follow a fixed pattern. We used a randomization mechanism so that the correct answer appears with a 0,5 probability. Each time users access such a question, the function returns a number that is displayed as the answer (Fig. 9).

Figure 9. The element that varies in the true/false questions.

3.1.4 Additional functionalities

For calculations involved in the application of the RSA algorithm, users can exploit a special calculator we developed particularly for exponentiation and modulo operations. We implemented this calculator as part of the application because sometimes it is more essential for learners to focus on the process itself than on performing arithmetic operations. By selecting the calculator icon from the navigation menu the calculator is displayed at the left side menu, as shown in Fig. 10.

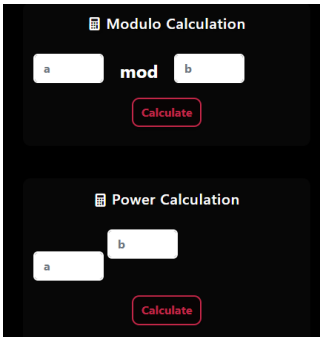

Figure 10. “RSA... made simple”: Calculator.

The calculator contains two sections: the upper section is for modulo operations while the lower section is for exponentiation operations. Exponentiation and modulo operations are used during all stages of the RSA algorithm, i.e., key computation, encryption and decryption of messages.

By clicking on the question mark icon in the navigation bar, users can quickly access a single pop-up screen (Fig. 11) containing the infographic and the description of each stage of the RSA algorithm included in detail in the “About” section. This feature allows users to recall the steps of RSA at a glance without having to navigate to other sections.

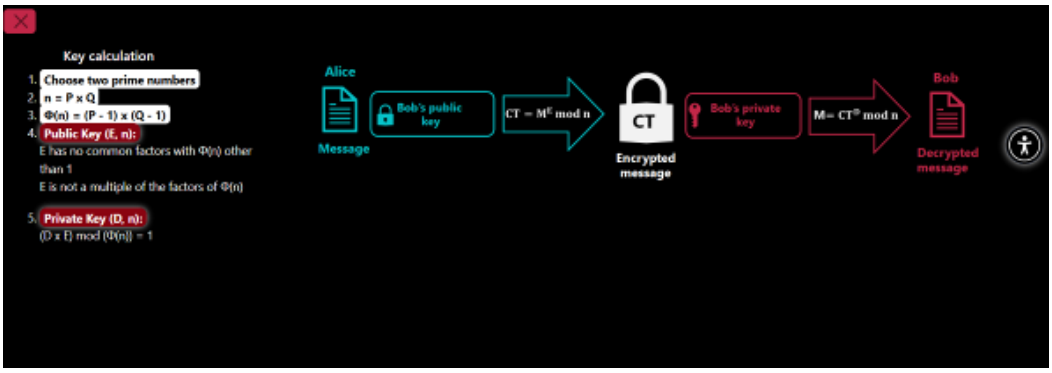

Figure 11. “RSA... made simple”: Infographic.

3.2 Development

The “RSA... made simple” application is an online tool we developed using various technologies to create a functional and modern solution. Initially, we chose React JS as the primary framework for building the user interface (UI). To handle all the computational and mathematical operations, we used JavaScript. For styling the screens, we combined the Bootstrap library with CSS to ensure the design is responsive across all screen sizes. We integrated the i18next library for multilingual support. Additionally, we incorporated the UserWay accessibility plugin to enhance the app's accessibility features. For development and code management, we used Visual Studio Code and GitHub, while hosting the application on GitHub Pages.

React JS [15] is a JavaScript library used for building web applications, widely known for its effectiveness in user interface development. It was introduced in 2013 and has since become the leading library for UI development. React uses components [17] self-contained units of code that include HTML, JavaScript, and CSS to build applications. This modular approach makes the code scalable, maintainable, and easy to read. A key feature of React is the Virtual DOM [8], which optimizes performance by updating only the necessary parts of the UI when changes occur. React also supports JSX (JavaScript XML) [13], allowing developers to write HTML-like syntax within JavaScript. The React ecosystem also uses hooks [18] and states [19] for managing variables, making the code more readable and maintainable. React was chosen for our project due to its suitability for applications requiring continuous user interaction and its flexibility for future expansion.

JavaScript [12] is the primary language for adding dynamic functionality to websites and web applications. It enables developers to create interactive features and modify content on web pages. We used JavaScript extensively to implement the core logic and operations of the RSA application. React Bootstrap [16], a popular open-source library, was employed for structuring the app and ensuring it is fully responsive on all screen sizes. CSS [7] was used for additional styling and layout customization, and we leveraged both inline styles and separate CSS files for better code organization.

The i18next [11] library provides multilingual support, allowing us to manage translations and format data such as dates and numbers according to the user's language or location. This ensures that our application is accessible in multiple languages. For the development environment, we utilized Visual Studio Code [21], which offers powerful code editing features such as auto-completion and error highlighting. GitHub [9] was used for version control, enabling us to track changes and collaborate effectively. GitHub Pages [10] served as our hosting solution, providing an easy and straightforward way to deploy the app. Lastly, UserWay’s accessibility plugin [20] was added to improve the app's usability, allowing users to adjust text size, word spacing, contrast, and more, ensuring better accessibility for all.

4 CONCLUSIONS AND FUTURE PLANS

With the intention to help interested users familiarize with the RSA algorithm due to its importance, its wide range of application and the observed difficulty among students regarding its understand, we designed and developed “RSA…made simple”, an educational web application with a graphical interface to demonstrate the operation and application of the RSA algorithm. Users can either experiment with their own instances or let the application perform key computation as well as encryption and decryption of messages provided in the form of numbers. Also, an evaluation section enables users to test their knowledge. “RSA…made simple” can be a useful tool particularly for students but also for other people who want to understand the RSA algorithm in the context of university courses on cryptography. Our application follows responsive design and supports accessibility features, and is currently available in greek and english. So far, we have received positive evaluation from students and faculty members as well as from other colleagues outside our academic community.

Future development of the application could focus on enhancing both functionality and educational value. One key improvement could involve extending the number range supported by the application, allowing for a broader set of RSA parameters to be explored enabling the application to handle more realistic instances. Additionally, support for encrypting simple phrases could be implemented by mapping characters to numerical values, applying RSA encryption to the resulting numerical representation, and decoding the output back into readable text. This approach would make the encryption process more tangible and relatable for users.

To further improve the learning experience, the application’s test module could be expanded with a greater variety of theoretical and practical questions. Introducing the option to simulate simple real-world scenarios, such as secure message exchange, would allow users to better grasp the purpose and mechanics of RSA. Moreover, the integration of other cryptographic algorithms, such as Diffie-Hellman, could facilitate comparative learning and deepen users’ understanding of different encryption methods and the challenges of cryptanalysis.

User engagement and feedback will also be a focus of future updates. A feedback mechanism - such as user reviews or a rating system - would not only highlight areas for improvement but also promote the application to a wider audience. Evaluating the application within academic settings, particularly in university-level cryptography courses, can offer practical insights for refinement and help establish it as a valuable educational resource.

REFERENCES

[1] T. Kritikos, E. Papaioannou, C. Kaklamanis, ”A web application for the visualization of parallel sorting algorithms,”. *Proceedings of the 16th annual International Conference on Education and New Learning Technologies (EDULEARN 24)*, IATED, pp. 600-609, 2024.

[2] C. Papastavrou, E. Papaioannou, C. Kaklamanis, ”"N2D": An online application for NFA-to DFA conversion and visualization,”. *Proceedings of the 17th annual International Technology, Education and Development Conference (INTED 23)*, IATED, pp. 6849-6858, 2023.

[3] R. L. Rivest, A. Shamir και L. M. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, pp. 120-126, 1978.

[4] B. Schneier, *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*. New York/NY: John Wiley & Sons, Inc., 1996.

[5] Z.S. Tramparis, E. Papaioannou, C. Kaklamanis, “FaRe: an online application for graphically designing Finite automata and computing equivalent Regular expressions,”. *Proceedings of the 15th annual International Conference of Education, Research and Innovation (ICERI 22)*, IATED, pp. 7445-7453, 2022.

[6] CrypTool - Online, Retreived from https://legacy.cryptool.org/en/cto/rsa-visual

[7] CSS, Retreived from https://www.w3schools.com/css/default.asp

[8] DOM, Retreived from https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[9] GitHub, Retreived from https://github.com/

[10] GitHub Pages, Retreived from https://pages.github.com/

[11] i18next, Retreived from https://www.i18next.com/

[12] JavaScript, Retreived from https://www.w3schools.com/js/

[13] JSX, Retreived from https://react.dev/learn/writing-markup-with-jsx

[14] Learn RSA Cryptography, G. Vatsan, Retreived from https://apps.apple.com/us/app/learn-rsa-cryptography/id1671454810

[15] React, Retreived from https://react.dev/

[16] React Bootstrap, Retreived from https://react-bootstrap.netlify.app/

[17] React Components, Retreived from https://www.w3schools.com/react/react_components.asp

[18] React Hooks, Retreived from https://legacy.reactjs.org/docs/hooks-overview.html

[19] React State, Retreived from https://legacy.reactjs.org/docs/state-and-lifecycle.html

[20] UserWay, Retreived from https://userway.org/about/

[21] Visual Studio Code, Retrieved from https://code.visualstudio.com/