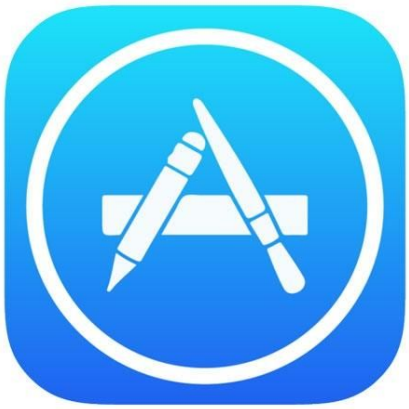
An iOS app which implements Encryption



Project Documentation

Anirudh Nagulapalli

Abstract:

This goal of this project is to design an iOS app which takes data from the user securely. The first step of the project is to provide user to set up an account with initial details (Example: Username, Email id, Phone number, password). When he/she sets them up, the user is asked which of the two (Phone number or Email) does he/she prefers to verify. Then a two-factor authentication number (Example: A random four-digit number) is sent to the respective phone number (or) the email. The user will have to verify by entering the number provided. When entered, and when it matches with the sent number, the passcode string and the two-factor authenticated number string is concatenated (Example: 'qwerty' + '9078' → qwerty9078) and encrypted using the MD5 Encryption or the RSA Encryption or the AES Encryption standards and sent to the other side.

The next page deals with the user providing private information like the License picture, Credit card information, Social Security Number, Health status, Bank account numbers etc which has to be on a secured connection. When the user finishes entering the required items, he can logout and the secured connection is no longer available with the server. If he has to login again, after entering the Email/Username and password he has to do the two-factor authentication again to access his profile.

Introduction:

Xcode is a set of integral programs that work together, used by software engineers and developers to write software for macOS, iOS(iPods, iPhones, iPads), the Apple Watch, and now the Apple TV. Xcode is a type of package called an IDE (Integrated Development Environment) with editors, compilers, and other software tools that work together to help us write software, compile it, load

onto a device, debug it, and ultimately submit it to the app store.

Xcode is a 100% free IDE that integrates all the tools which are required. It is the only program that helps us to make native Apple product applications. This robust interface takes us from composing source code, to the process of debugging and even designing next user interface and uploading to the App Store all within your window.

The workspace in Xcode helps us to compile errors or issues. We can test our iOS apps either in the simulator that comes with Xcode, or on our own physical iOS device. If you are developing a Mac OS X app, you just run it on your Mac. It can also easily sign and organize your Mac or a iOS app and straight way submit to the App Store through iTunes Connect.

Background:

Xcode is an integrated development environment for macOS containing a suite of software development tools developed by Apple for developing software for macOS, iOS, watchOS and tvOS. [1] With the help of this software development tool, an app can be created which runs on macOS or iOS or watchOS or tvOS or on all devices together as such. Even though Xcode supports source code for programming languages like C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez), and Swift, it only take two platforms as base. One being Objective C, the project is done using the second platform Swift.

Swift is a general-purpose programming language built using a modern approach to safety, performance, and software design patterns. The goal of the Swift project is to create the best available language for uses ranging from systems programming, to mobile and desktop apps, scaling up to cloud services. [2] The real time use of this application is in the field of secure information sharing system.

The main objective is to provide the user, a form to fill in his personal information so that he can send them through a secured way. To achieve this, the data in the form of pure text is converted into something which is not easily readable and is sent over a channel.

Methodology:

The idea behind this application is to provide a secure channel between any two users. The actual goal was to provide doctors, a way to encrypt their personal information like (Medical license numbers, expiration dates etc) and send them through a secured channel.

The main method to convert the data which is in the form of a normal text (Plain text) to something which is unreadable is achieved by encryption. In cryptography, encryption is the process of encoding a message or information in such a way that only authorized parties can access it. [3] To perform encryption, the necessary inputs are the original text (Plain text) and a key. In cryptography, a key is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm. [4] Out of the two types of cryptography, we use the public key cryptosystem is a system where a single key is used for both

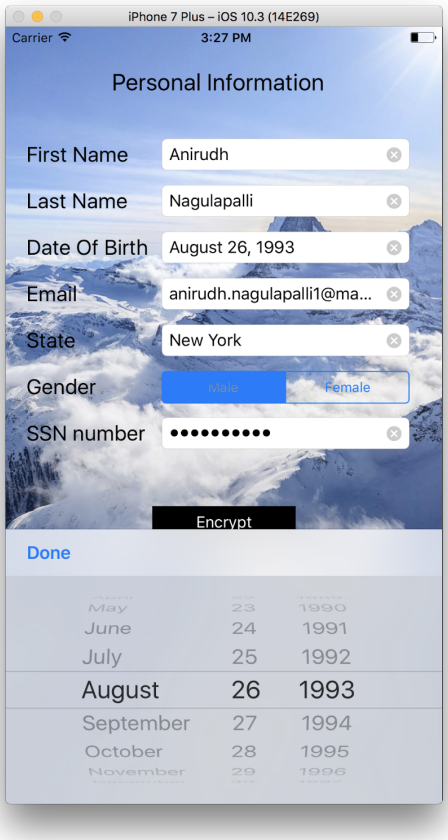
encryption and decryption. A sender and a recipient must already have a shared key that is known to both. []

Experiments:

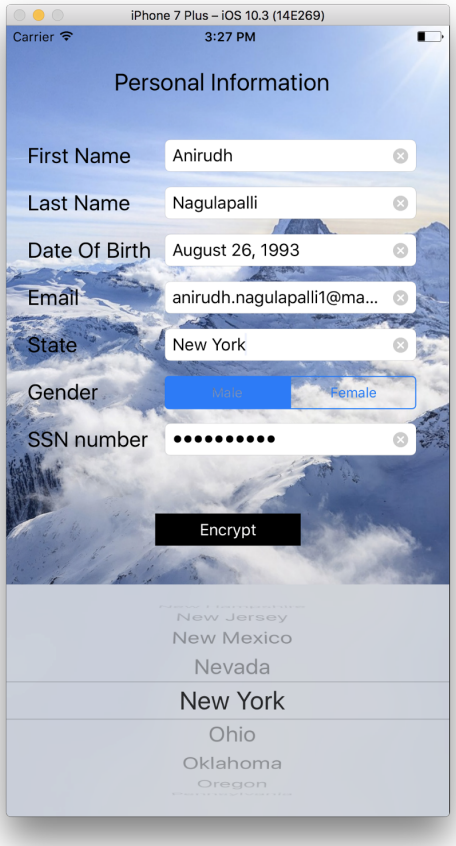
The main set up of the application consists of building a neat and clean UI for the user. The basic details for the user to enter are First name, Last name, an Email id (for authentication if required), a Phone number (for authentication if needed), a password to set up the account, Date of Birth to verify if he/ she comes under the category, Gender, State and finally the Social Security Number which should not come anywhere into the hands of masquerades.

The setup consists of simple Labels, Textfields, Date Picker, UI Picker, Segmented control, Secure text entry fields and some Buttons. The Labels are used to specify what the respective textfields show, the textfields are for the user to input the data according to the specified label for the respected textfield.

The Date Picker is used generally to specify the date and time of the event. Here in this project, we use only the date version of the date picker to represent the users Date of Birth. The user has to scroll through the input and select an Year, Month and Date.

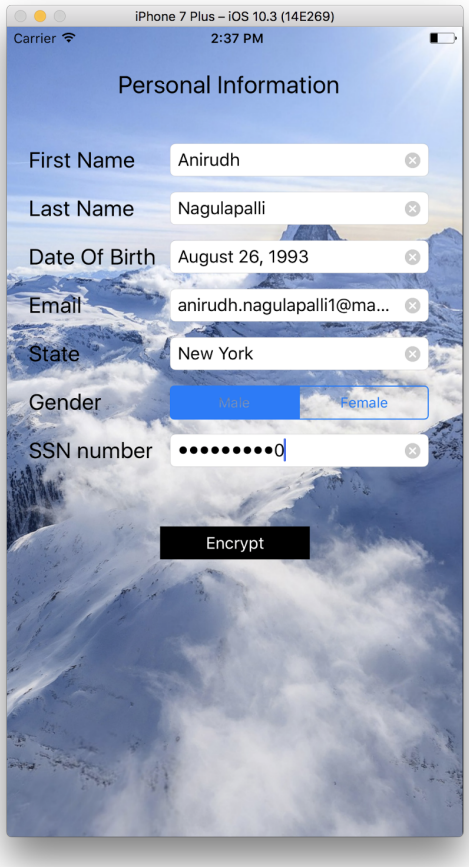


The UI Picker is similar to the Date Picker but the user has to select from a list of given options. Here in this app the user is asked to select the state he/she's from the list of 50 states in USA. A secure text entry field is used to enter the Social Security number. These fields are mapped into variables using the 'View Controller'. The ideology is to encrypt the SSN number along with the details using a pre determined key and sending it over a channel securely. While decrypting, the same key is used to decrypt the Ciphertext.



Analysis:

The app with the needed UI configurations has been set up. The required elements are dragged from the Object library and dropped into the workspace area. Creating Stacks helps to arrange the labels as a whole and textfields as a whole. After arranging the elements, it looks different in different schemas. Like for example when compared with an iPhone 7 plus schema with an iPhone 4S schema, the elements don't fit in as naturally. Hence, constraints are to be added for each and every element. Setting Constraints are nothing but a way of telling the elements where to start and where to end in a schema. The different types of constraints include Background Constraints (Background top, Background bottom, Background leading, Background trailing), Label Constraints (Label top, Label center), Stack View Constraints (Stack View top, Stack View bottom, Stack View leading, Stack View trailing).



Conclusion:

Though the process of setting up the User Interface of the app is more fun and creative, the actual warnings appear at the time of coding for encryption. In this app the standard procedure of Advanced Encryption Standard of 128 bit key length is being used. The whole process of setting up w_{new} , Shifting, S.Box, XOR operation of the first value with R.Con and finally $w(j) = w(j-4) \oplus w_{new}$ can be programmed in Swift platform. Orelse, a library called 'CryptoSwift', an open

source library can be used to achieve AES-128 or AES-192 or even AES-256 encryption. The other advantage of the library include the use of a Hash digest, Cyclic Redundancy Check, Authentication (using HMACs or SHA256) and Data Padding. [5]

Even though confidentiality is achieved, integrity and authenticity still lacks. So that can be provided by using private key cryptosystem to some extent and hashing.

Future Aspects:

With the use of Symmetric Cryptosystem, a single key is used on both ends of encryption and Decryption. There is more chance of Man in the Middle attack using Symmetric Cryptosystem. Hence the use of Asymmetric Cryptography while help in a ransom amount. In Asymmetric Cryptosystem there will be a public key and a private key for every user. The public key is set to be seen for everybody but the private key should be used to decrypt the content. Another thing which can be done is using hashing. Hashing enables the to bring authenticity to the application. It ensures that the plain text is not changed by the 'Man in the middle' in anyway.

Acknowledgments:

Special thanks and recognition goes to our advisor, Professor. Dr. Pablo Rivas who guided, inspired and motivated me through this study. I always wanted to build an application of my own. But building on what has been studied on for a semester gave me a lot of excitement and self satisfaction. Also, I would like to thank the Marist College for giving us an opportunity to get introduced to this course work.

References:

1. https://en.wikipedia.org/wiki/Xcode
2. https://swift.org/about/
3. https://en.wikipedia.org/wiki/Encryption
4. https://en.wikipedia.org/wiki/Key_(cryptography)
5. https://github.com/krzyzanowskim/CryptoSwift
6. https://www.synopsys.com/software-integrity/resources/knowledge-database/cryptography.html