UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Elliptic Curve Cryptography Services for Mobile Operating Systems

Ana Raquel Correia Figueira

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro Ricardo Morais Inácio

Covilhã, Outubro de 2015

To my loving family, who guided me to where I am today.

Acknowledgments

First and foremost, I must thank my family, who have always been extremely supportive of all my decisions. To my parents, Ana Maria and António, for everything they have done to be able to provide me with an education, and for everything that they have taught me. To my sister Sónia and to Francisco for their support and to their new addition to the family, my adorable nephew Kiko.

I want to thank my friends, old and new, for their help throughout this phase of my life, which was filled with huge changes and they certainly gave me the strength to accept new challenges and conciliate work with studying. Thanks to you all, in particular to Ana Costa, whose help and support was immeasurable.

I would like to thank my uncle Jorge Correia, my aunt Anabela Correia and to my cousin Ricardo Correia for all the support they gave me and for temporarily welcoming me in their home. Thank you all for teaching me so much about life.

Last, but not least, I want to thank my supervisor, Professor Dr. Pedro Inácio, for his guidance in this dissertation and for always being available especially when I was living far and could not always come to meet him in person. His commitment to his students is something out of the ordinary and I truly appreciate it.

Resumo

Dispositivos móveis como computadores portáteis, *smartphones* ou *tablets*, são, nos dias de hoje, considerados objectos indispensáveis pela grande maioria das pessoas residentes em países desenvolvidos. Por serem utilizados como assistentes pessoais ou de trabalho, alguns destes dispositivos guardam, processam e transmitem dados sensíveis ou privados. Naturalmente, o número de aplicações móveis com mecanismos criptográficos integrados ou que oferecem serviços de segurança, tem vindo a aumentar de forma significativa nos últimos anos. Infelizmente, nem todas as aplicações são seguras por construção, e outras podem não implementar as primitivas criptográficas corretamente. Mesmo aquelas que as implementam corretamente podem sofrer de problemas de longevidade, já que primitivas criptográficas que são hoje em dia consideradas seguras podem tornar-se obsoletas nos próximos anos. O Rivest, Shamir and Adleman (RSA) constitui um exemplo de um sistema criptográfico muito popular que se pode tornar obsoleto a curto prazo. Enquanto que os problemas de segurança em ambientes de computação móvel podem ser de média severidade para utilizadores casuais, estes são normalmente críticos para várias classes profissionais, nomeadamente advogados, jornalistas e oficiais da justiça. É, por isso, importante, abordar estes problemas de uma forma estruturada.

Este programa de mestrado foca-se na engenharia e implementação de uma aplicação móvel que oferece uma série de serviços de segurança. A aplicação foi desenhada para ser segura por construção para o sistema operativo *Windows Phone* 8.1 que, altura em que esta dissertação foi escrita, era a plataforma com a oferta mais discreta em termos de aplicações deste tipo. A aplicação fornece funcionalidades como trocar um segredo criptográfico entre duas entidades de forma segura, cifra, decifra e assinatura digital de mensagens e ficheiros, gestão de contactos e chaves de cifra, e geração e armazenamento seguro de palavras-passe. Parte das primitivas criptográficas utilizadas neste trabalho fazem parte da teoria da criptografia em curvas elípticas, para a qual se acredita que o problema do logaritmo discreto é de mais difícil resolução e para o qual a manipulação de chaves é mais simples. A biblioteca que define uma série de curvas, e contendo os procedimentos e operações que suportam as primitivas criptográficas, foi totalmente implementada no âmbito deste trabalho, dado ainda não existir nenhuma disponível no seu início, compreendendo assim uma das suas contribuições. O trabalho evoluiu da análise do estado da arte para o levantamento dos requisitos e para a fase de engenharia de software, aqui descrita detalhadamente, culminando no desenvolvimento de um protótipo. A engenharia da aplicação incluiu a definição de um sistema de confiança para troca de chaves públicas e também modelação da base de dados de suporte.

Os resultados mais visíveis deste programa de mestrado são o protótipo da aplicação móvel, completamente funcional e disponibilizando as funcionalidades de segurança acima mencionadas, a implementação de uma biblioteca Elliptic Curve Cryptography (ECC) para *framework* .NET, e esta dissertação. O código fonte com a implementação da biblioteca foi publicada *online*. O seu desenvolvimento e melhoramento foi sobretudo dominado por testes unitários. A biblioteca e a aplicação móvel foram desenvolvidas em C♯. O nível de segurança oferecido pela aplicação é garantido através da orquestração e combinação de algoritmos da criptografia de chave simétrica atuais, como o Advanced Encryption Standard (AES) e o Secure Hash Algorithm 256 (SHA256), com as primitivas ECC. A geração de palavras-passe é feita recorrendo utilizando vários sensores e dispoitivos de entrada como fontes de entropia, que posteriormente são alimentadas a uma função de *hash* criptográfica. As palavras-passe são guardadas numa base de dados cifrada, cuja chave de cifra muda sempre que a base de dados é aberta, sendo obtida através da aplicação de um Password-Based Key Derivation Function 2 (PBKDF2) a uma palavra-passe mestre. O modelo de confiança para chaves públicas desenhado no âmbito deste trabalho é inspirado no Pretty Good Privacy (PGP), mas a granularidade dos níveis de confiança é superior.

Palavras-chave

Aplicação Móvel, Assinatura Digital, Criptografia de Chave Pública, Criptografia de Chave Simétrica, Criptografia em Curvas Elípticas, Gestão de Palavras-Passe, Windows Phone, Segurança,

Extended Abstract in Portuguese

Introdução

Este capítulo, escrito em língua Portuguesa, pretende resumir o corpo desta dissertação, de uma forma mais alargada do que no resumo. Começa por enquadrar a dissertação, analisando as diferentes aplicações actualmente disponíveis, as tecnologias usadas e falando sobre alguns pressupostos actuais da criptografia. Depois segue para a análise de requisitos e dos diagramas de engenharia de software. São depois explicados quais os serviços criptográficos que a aplicação deverá suportar. Os detalhes de implementação da aplicação são abordados adiante. Por último, são apresentadas as principais conclusões relativas ao trabalho aqui apresentado, assim como algumas direcções para trabalho futuro.

Nos capítulos escritos em língua Portuguesa, os acrónimos que são utilizados aparecem nas suas formas longas, apesar das suas formas curtas serem mantidas com o respectivo acrónimo em língua Inglesa para efeitos de consistência. Os acrónimos cuja tradução não seja directamente possível, devido a se referirem a serviços ou aplicações, aparecem em itálico como estrangeirismos.

Enquadramento, Descrição do Problema e Objetivos

A evolução dos telefones tem sido exponencial, e enquanto que inicialmente eram apenas utilizados para efectuar chamadas ou enviar mensagens, hoje em dia têm funcionalidades muito mais avançadas. Actualmente, e inerente às suas funcionalidades, os *smartphones* contêm informação sobre os utilizadores considerada privada e sensível. Esta informação inclui dados sobre o utilizador, os seus contactos, mensagens, emails, fotografias, vídeos e muitos outros tipos de ficheiros. Isto leva a que hajam sérios problemas de segurança, privacidade e confidencialidade, pois normalmente a informação não é guardada de forma segura e está vulnerável a possíveis ataques.

Existe uma clara necessidade de aplicações criptográficas que permitam tratar informação confidencial de forma a prevenir possíveis fugas. Estas aplicações são particularmente importantes para algumas classes profissionais como as de jornalistas, forças de segurança e outras pessoas que lidem com questões de justiça ou governamentais. Na realidade, qualquer organização, instituição ou pessoa que manipule informação privada e confidencial precisaria de um sistema assim.

Alguns especialistas acreditam que o algoritmo *Rivest, Shamir and Adleman (RSA)* poderá ser quebrado nos próximos anos [Tom13]. É aconselhável que o tamanho das chaves seja aumentado, embora actualmente já seja bastante grande. É sugerido que o *RSA* seja substituído por outros sistemas criptográficos e isto levou a um novo interesse na criptografia sobre curvas elipticas, cujos tamanhos de chave são muito inferiores e é considerada criptograficamente mais forte.

O objectivo deste programa de mestrado foi o de criar uma aplicação de segurança para dispositivos móveis, mais concretamente para o sistema operativo *Windows Phone 8.1*. Na altura em que esta dissertação foi escrita não foi encontrada nenhuma aplicação para o sistema operativo escolhido com as funcionalidades desejadas. As funcionalidades identificadas como importantes e necessárias incluem a criação de um sistema de troca de chaves de confiança, cifragem e decifragem de mensagens e ficheiros. Por último, o típico utilizador tem várias *passwords* e torna-se dificil memorizá-las todas, portanto existe a necessidade de armazenar *passwords* de forma segura. Para atingir os objectivos aqui propostos, as seguintes fases foram estabelecidas:

1. A primeira fase consiste em contextualizar o problema e estudar os *standards* actuais de sistemas e esquemas criptográficos que são usados, bem como a teoria sobre criptografia em curvas elípticas. Nesta fase é também necessário efectuar uma pesquisa nas lojas de aplicações dos diferentes sistemas operativos por aplicações com funcionalidades semelhantes às implementadas neste programa de mestrado;

2. A segunda fase implica o estudo e a identificação dos mecanismos criptográficos adequados para este trabalho, bem como a definição detalhada das funcionalidades a serem implementadas;

3. A terceira fase inclui todo o processo de engenharia de software da aplicação. Começa pela identificação dos requisitos e casos de uso e termina com os diferentes diagramas que representam os casos de uso, actividades e componentes.

4. A quarta fase consiste na implementação da aplicação e das bibliotecas necessárias para a inclusão das primitivas de criptografia sobre curvas elípticas. Inclui também detalhes sobre os testes da aplicação em emuladores e dispositivos físicos.

5. A última fase inclui melhorias na aplicação, voltando à fase de engenharia de software, caso seja necessário.

Principais Contribuições

Esta secção descreve brevemente as principais contribuições que resultaram do trabalho de investigação aqui apresentado.

Este programa de mestrado é motivado principalmente pela falta de serviços de segurança em *smartphones*. A principal contribuição é a construção de um protótipo de uma aplicação de segurança para o sistema operativo *Windows Phone* 8.1. À data de entrega desta dissertação, esta é a aplicação de segurança mais completa para este sistema operativo.

Outra contribuição é o desenvolvimento de um esquema de gestão de chaves que associa contactos e chaves públicas, e que define a estrutura de troca de chaves e da confiança nas chaves. A estrutura de confiança é inspirada nos esquemas *Public Key Infrastructure (PKI)* e *Pretty Good Privacy (PGP)*. A engenharia de software e os detalhes de implementação da aplicação são vistos como sendo secundários, mas importantes contribuições deste trabalho, pois podem ajudar futuros programadores no processo de criar aplicações seguras. Além da aplicação desenvolvida, também foi criada uma biblioteca com as funções criptográficas. A mesma está disponível de forma gratuita para que possa ser utillizada por outros programadores, sob a licença *Massachusetts Institute of Technology (MIT)* [Ope15].

Este trabalho foi também submitido no concurso Open Mind ESEGUR 2014 onde foi apresentado e ficou entre os semi-finalistas.

Aplicações Relacionadas, Tecnologias e Preliminares em Criptografia
Nesta secção é discutido o sistema operativo para que esta aplicação foi criada e é feita uma comparação com os restantes sistemas operativos para dispositivos móveis. Os preliminares que fazem parte da base deste trabalho também são aqui discutidos.

Foram feitas pesquisas nas diferentes lojas de aplicações dos principais sistemas operativos para dispositivos móveis: Windows Phone, iOS e Android. Os resultados obtidos vão ser agora explicados.

Android é um sistema operativo *open-source* com base no *kernel* Linux, desenvolvido pela *Google*. Actualmente está instalado em mais de 78% dos dispositivos móveis e é o lider de mercado neste segmento. Após a pesquisa efectuada na loja oficial de aplicações Android (Google Play Store [Goo15]), a conclusão foi que não existem muitas aplicações que usam *Elliptic Curve Cryptography (ECC)* em Android. Existem muitas aplicações que permitem troca de mensagens de forma segura, mas as mesmas utilizam *RSA*. As poucas aplicações encontradas que usam realmente *ECC* têm funcionalidades muito limitadas e apenas permitem a cifragem e decifragem de texto.

O iOS é um sistema operativo proprietário desenvolvido pela Apple Inc. e que é usado em dispositívos como iPhone, iPad e iPod e Apple TV. O iOS tem a segunda maior fatia do mercado, após o Android, actualmente com 18% do mercado. Foi efectuada uma pesquisa também na loja de aplicações oficial da Apple [App15], segundo o critério de encontrar aplicações que ofereçam funcionalidades semelhantes às desejadas para aplicação a ser criada para esta dissertação. Não foram encontradas aplicações de segurança que utilizam *ECC* na sua implementação.

Windows Phone 8.1 é um sistema operativo desenvolvido para *smartphones* pela Microsoft. Este foi o sistema operativo escolhido para desenvolver a aplicação desta dissertação. A pesquisa efectuada na loja oficial do Windows não teve qualquer resultado, o que aumentou a motivação para a criação de uma aplicação com estas funcionalidades para esta plataforma.

Engenharia de Software

Foi feita uma análise dos requisitos do ponto de vista de funcionalidades. Os requisitos funcionais identificam tarefas, acções e actividades tais como: geração de palavras-passe e segredos criptográficos baseados em verdadeira aleatoriedade, geração de pares de chaves sobre curvas elípticas, troca de chaves secretas entre dois utilizadores, cifragem de mensagens com possibiliade de exportação para criptograma via email ou *Short Message Service (SMS)*, decifragem de mensagens, assinatura digital de ficheiros através de *Elliptic Curve Diffie-Hellman (ECDH)*, importação, exportação e manuseamento de chaves públicas, autenticação através de criptografia de chaves públicas, codificação e descodificação de mensagens, armazenamento seguro de ficheiros no dispositivo e gestão de palavras-passe. Como requisitos não funcionais, a aplicação deve ser intuitiva e de fácil compreensão; a aplicação deve ter um *layout* simples e categorizado, o tempo de resposta não deve ser superior a 1 segundo para funções normais e 2.5 para funções criptográficas.

Foram apresentados casos de uso para as funcionalidades previstas para a aplicação. No diagrama da figura 3.1 é ilustrado o caso de uso geral da aplicação. Na figura 3.2 ilustra-se o subsistema de gestão de palavras-passe, onde é possível adicionar, editar ou apagar uma palavra-passe, sendo que, para adicionar ou editar o utilizador pode escolher fazê-lo através do gerador de palavras-passe providenciado pela aplicação ou introduzindo-a manualmente. No diagrama de gestão de chaves, representado na figura 3.3, o utilizador pode gerar, editar ou apagar chaves previamente armazenadas. A figura 3.4 ilustra os casos de uso na gestão de contactos, onde é possível adicionar novos contactos, importar as suas chaves e editar ou apagar contactos já existentes. No subsistema de gestão de mensagens é possível cifrar, decifrar e apagar mensagens armazenadas. Na gestão de ficheiros é possível fazer a cifragem e decifragem dos mesmos.

Para a concretização desta aplicação, foi desenhada uma base de dados para armazenar a informação necessária. Na figura 3.7 está representado o modelo de entidade-relacionamento desta base de dados. O modelo relacional desta base de dados, representado na figura 3.8, ilustra mais detalhadamente as tabelas criadas e os atributos de cada entidade.

Os diagramas de actividade são representações gráficas do fluxo de acções e actividades e são usados para mostrar o fluxo de controlo geral. Na figura B.2 é ilustrado o processo de cifragem de um ficheiro; no diagrama da figura B.1 está representado o processo de adicionar uma palavra-passe; o processo de partilha de chaves é apresentado na figura B.3; no diagrama da figura B.4 está ilustrado como se importa uma chave.

A figura 3.9 representa esquematicamente como os três componentes se interligam por forma a permitir o bom funcionamento da aplicação. Estes três componentes são: a aplicação móvel, a biblioteca partilhada para a lógica da aplicação e a biblioteca `ECCryptoLib` (onde se encontra a lógica criptográfica da aplicação).

Segurança e Serviços Criptográficos

Os serviços criptográficos identificados para implementar na aplicação foram os seguintes: geração de palavras-passe e segredos criptográficos baseados em fontes de verdadeira aleatoriedade, disponíveis nos dispositivos móveis; geração de pares de chaves sobre curvas elípticas; criação de chaves secretas entre dois ou mais utilizadores; cifragem e decifragem de mensagens de texto; assinatura digital de ficheiros através do algoritmo de assinatura digital sobre curvas elípticas; importação, exportação e gestão de chaves públicas; autenticação através de criptografia de chaves públicas; codificação e descodificação de mensagens; armazenamento seguro de ficheiros no dispositivo; e gestão de palavras-passe.

De modo a ter as funções criptográficas necessárias, foi criada uma biblioteca onde foi implementada toda a lógica de criptografia necessária. Esta biblioteca pode facilmente ser importada pela aplicação `ECCrypto`, bem como por outras aplicações que necessitem destes serviços. Entre as funções implementadas estão: geração e validação de palavras-passe; geração de pares de chaves sobre curvas elípticas, de acordo com a curva escolhida, baseados em fontes de aleatoriedade do *smartphone*; funções de *ECDH* necessárias para troca de chaves de forma segura; cifragem e decifragem de mensagens e ficheiros; algoritmo de assinatura digital sobre curvas elípticas; e os algoritmos de *hash* necessários.

Foi criado um sistema de gestão de confiança para garantir a confiança das chaves dos contactos. O nível de confiança é uma percentagem que depende do número de contactos que confirmam essa mesma chave. Por defeito, são necessárias três confirmações de modo a ter uma chave completamente confiável, mas este número pode ser configurado nas definições da aplicação.

O sistema operativo Windows Phone 8.1 utiliza componentes de *hardware* baseados nos mais recentes *standards* de segurança, que servem para proteger tanto os dispositivos como a informação neles contida. Partindo do princípio que a segurança destes componentes é garantida, é possível afirmar que a aplicação é segura. Algumas das restrições do sistema operativo, que surgiram inicialmente como constragimentos no desenvolvimento da aplicação, acabaram por aumentar a segurança desta. De modo a reforçar ainda mais a segurança da própria aplicação, é aconselhado ao utilizador definir uma palavra-passe de bloqueio.

Implementação e Testes

A biblioteca `ECCryptoLib` foi implementada por não existir nenhuma disponível simultaneamente para ambas as plataformas pretendidas. Foram adicionadas à biblioteca as funções de hash *Secure Hash Algorithm (SHA)-1*, *SHA-256*, *SHA-384*, *SHA-512* e *Message-Digest 5 (MD5)*. Foram também incluídas funções *Hash-based Message Authentication Code (HMAC)-SHA-1*, *HMAC-SHA-256*, *HMAC- SHA-384*, *HMAC- SHA-512* e *HMAC- MD5*. Foram adicionadas as funções de cifragem e decifragem utilizadas pelo algoritmo *Advanced Encryption Standard (AES)*. A biblioteca contém um gerador de palavras-passe e métodos que avaliam a força destas. Foi também incluída uma classe onde se encontram as curvas implementadas.

Para testar o protótipo da aplicação foram utilizados os diferentes emuladores disponíveis no *Visual Studio* e três *smartphones* de diferentes gamas. Durante a implementação da biblioteca, dada a necessidade de ir testando os métodos nela implementados, criou-se um projecto `ECCryptoLibTest`. Foram feitos testes unitários à biblioteca.

Quando terminada a aplicação, foi submetida para a Loja do Windows Phone através do *dashboard* disponibilizado para o efeito. A aplicação pode ser descarregada da Loja do Windows Phone, em todo o mundo, de forma gratuita. A biblioteca foi também disponibilizada no *GitHub* como repositório público, onde pode ser melhorada por outros utilizadores que queiram contribuir.

Conclusões e Trabalho Futuro

Os objectivos inicialmente propostos neste trabalho foram alcançados e resultaram na implementação do `ECCrypto`, que é uma aplicação desenvolvida para *Windows Phone*, que procura garantir a segurança e confidencialidade na comunicação entre dois ou mais utilizadores. Esta aplicação utiliza funções criptográficas mais fortes do que as tipicamente usadas, e agrega diversas funcionalidades como armazenamento, geração e gestão de palavras-passe, cifragem e decifragem de mensagens e ficheiros, envio de *SMS* e *emails* cifrados, geração e gestão de chaves, gestão de contactos e chaves públicas, troca de chaves através de códigos *Quick Response (QR)*, sistema de gestão de confiança e assinatura digital de mensagens e ficheiros.

As funções de *ECC* foram implementadas de raíz, pois aquando do desenvolvimento deste trabalho não existiam bibliotecas com as funções necessárias. Assim, estas funções, bem como as operações, foram implementadas. O mecanismo de geração de pares de chaves assimétricas, métodos para assinatura digital, funções de cifragem e decifragem de mensagens e ficheiros utilizando AES, um gerador de palavras-passe aleatório e vários algoritmos de hash foram também incluídos.

Dada a enorme importância e valor da informação nos dias de hoje, a aplicação `ECCrypto` aparece como uma alternativa de confiança para o utilizador que pretende manter a sua informação e comunicação seguras. Como trabalho futuro, a aplicação deveria ser testada quanto à usabilidade devido à sua dimensão e complexidade. É também necessário garantir a retenção de utilizadores através das ferramentas apropriadas. Seria interessante incluir outro tipo de curvas elípticas *standard*, para que os utilizadores não ficassem limitados a curvas *National Institute of Standards and Technology (NIST)*. A implementação noutros sistemas operativos é também um ponto muito importante, uma vez que permitirá a comunicação entre pessoas que utilizam diferentes plataformas. O *upgrade* para o *Windows* 10 será igualmente importante, pois permitirá a utilização da aplicação nos vários dispositivos Windows. Outra funcionalidade que poderá vir a ser implementada é a sincronização entre os vários dispositivos de um mesmo utilizador.

Abstract

Mobile devices as smartphones, tablets and laptops, are nowadays considered indispensable objects by most people in developed countries. As personal and work assistants, some of these devices store, process and transmit sensitive and private data. Naturally, the number of mobile applications with integrated cryptographic mechanisms or offering security services has been significantly increasing in the last few years. Unfortunately, not all of those applications are secure by design, while other may not implement the cryptographic primitives correctly. Even the ones that implement them correctly may suffer from longevity problems, since cryptographic primitives that are considered secure nowadays may become obsolete in the next few years. Rivest, Shamir and Adleman (RSA) is an example of an widely used cryptosystem that may become depleted shorty. While the security issues in the mobile computing environment may be of median severity for casual users, they may be critical for several professional classes, namely lawyers, journalists and law enforcement agents. As such, it is important to approach these problems in a structured manner.

This master's program is focused on the engineering and implementation of a mobile application offering a series of security services. The application was engineered to be secure by design for the Windows Phone 8.1 Operating System (OS) which, at the time of writing this dissertation, was the platform with the most discreet offer in terms of applications of this type. The application provides services such as secure exchange of a cryptographic secret, encryption and digital signature of messages and files, management of contacts and encryption keys and secure password generation and storage. Part of the cryptographic primitives used in this work are from the Elliptic Curve Cryptography (ECC) theory, for which the discrete logarithm problem is believed to be harder and key handling is easier. The library defining a series of curves and containing the procedures and operations supporting the ECC primitives was implemented from scratch, since there was none available, comprising one of the contributions of this work. The work evolved from the analysis of the state-of-the-art to the requirements analysis and software engineering phase, thoroughly described herein, ending up with the development of a prototype. The engineering of the application included the definition of a trust model for the exchange of public keys and the modeling of the supporting database.

The most visible outcomes of this master's program are the fully working prototype of a mobile application offering the aforementioned security services, the implementation of an ECC library for the .NET framework, and this dissertation. The source code for the ECC library was made available online on GitHub with the name `ECCryptoLib` [Ana15]. Its development and improvement was mostly dominated by unit testing. The library and the mobile application were developed in C$\sharp$. The level of security offered by the application is guaranteed via the orchestration and combination of state-of-the-art symmetric key cryptography algorithms, as the Advanced Encryption Standard (AES) and Secure Hash Algorithm 256 (SHA256) with the ECC primitives. The generation of passwords is done by using several sensors and inputs as entropy sources, which are fed to a cryptographically secure hash function. The passwords are stored in an encrypted database, whose encryption key changes every time it is opened, obtained using a Password-Based Key Derivation Function 2 (PBKDF2) from a master password. The trust model for the public keys designed in the scope of this work is inspired in Pretty Good Privacy (PGP), but granularity of the trust levels is larger.

Keywords

Digital Signature, Elliptic Curve Cryptography, Microsoft Windows Mobile, Mobile Application, Password Management, Public Key Cryptography, Trust Model, Security, Symmetric Key Cryptography

Contents

List of Figures

List of Tables

Acronyms

3DES	Triple Data Encryption Standard
ACM	Association for Computing Machinery
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
AOSP	Android Open Source Project
ASLR	Address Space Layout Randomization
BIOS	Basic Input/Output System
CCA	Chosen Ciphertext Attack
CLI	Common Language Infrastructure
CPA	Chosen Plaintext Attack
CSS	Cascading Style Sheets
DAP	Discrete Logarithm Problem
DEP	Data Execution Prevention
DES	Data Encryption Standard
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECES	Elliptic Curve Encryption Scheme

ECIEDS	Elliptic Curve Integrated Encryption Scheme
ECIES	Elliptic Curve Integrated Encryption Scheme
FIPS	Federal Information Processing Standards
GPS	Global Positioning System
HMAC	Hash-based Message Authentication Code
HTML	HyperText Markup Language
ID	Identifier
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IM	Instant Messaging
IPsec	Internet Protocol Security
KDF	Key Derivation Function
LINQ	Language Integrated Query
LTE	Long-Term Evolution
MAC	Message Authentication Code
MD5	Message-Digest 5
MIT	Massachusetts Institute of Technology
MMS	Multimedia Messaging Service
NIST	National Institute of Standards and Technology
NSA	National Security Agency
OS	Operating System
OTP	One Time Password

PBKDF2	Password-Based Key Derivation Function 2
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
QR	Quick Response
RC	Rivest Cipher
RFC	Request for Comments
RSA	Rivest, Shamir and Adleman
SD	Secure Digital
SDK	Software Development Kit
SDL	Security Development Lifecycle
SEC	Standards for Efficient Cryptography
SECG	Standards for Efficient Cryptography Group
SHA	Secure Hash Algorithm
SHA256	Secure Hash Algorithm 256
SIP	Session Initiation Protocol
SLAT	Second Level Address Translation
SMS	Short Message Service
SQL	Structured Query Language
SSH	Secure Shell
SRTP	Secure Real-time Transport Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module

UBI	Universidade da Beira Interior
UEFI	Unified Extensible Firmware Interface
UI	User Interface
UML	Unified Modeling Language
VPN	Virtual Private Network
XAML	Extensible Application Markup Language
XAML	eXtensible Application Markup Language
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language for Transformation

Chapter 1

Introduction

This dissertation describes the work on the design and implementation of an application for mobile devices offering several cryptographic related functions/services. The scope and motivation of the dissertation are described in the following sections, along with the problem statement and objectives, the adopted approach for solving the problem, the main contributions, and the dissertation overview.

1.1 Scope and Motivation

Phones have seen an exponential evolution since they first appeared in the nineteen nineties. Back then, their usage was limited to making calls and sending text messages. With the evolution of technology, the capabilities of phones have grown and we now live in the era of smartphones with advanced computing and connectivity functionalities. Smartphones are part of the life of nearly everyone in developed countries, carrying information that is both personal and confidential. They are used to store private information about the users and their contacts, such as messages, e-mails, pictures, videos and multiple other types of files.

Even with all the evolution of technology and of smartphones, serious security issues exist. E.g., text messages can be intercepted and the content can be obtained by third parties and the same problem applies to e-mails. While users expect for their messages and e-mails to be private and only seen by the recipient of the message, the policies of the companies hosting the service may disregard the need for privacy and confidentiality. Insecurity applies to both communications and storage of information on mobile devices or computers, where files are, most of the times, kept unencrypted or without any other security assurances beyond the basic ones provided by the Operating Systems (OSs) or hardware.

On the other hand, most applications and implementations for the different mobile platforms and OSs are not reliable from a security perspective [J.M15], and some applications offering security services have begun to arise in the software markets, so as to fight this tendency. However there are not many applications that incorporate a large set of security functionalities in a single application.

Security awareness has never been so present in our day-to-day lives, emphasizing the importance of this subject nowadays. It has been on the news for the best part of the last couple of years, evidenced by the whistle-blower Edward Snowden. This brought the problem of lack of privacy and confidentiality to the public opinion.

All these factors corroborate the belief that security must be thought of from the stage of software engineering in order to guarantee it (security by design). Everything, from the way people communicate, manage secrets needs to be reconsidered and improved in order to be truly useful to the user without becoming a hassle.

Security applications for mobile devices are needed for various social classes such as journalists and security forces, as well as people who deal with issues of justice or government. In fact, any organization or institution who handles any private and confidential information would require such a system, in order to prevent possible leaks of information.

These are revolutionary times for cryptography since it is believed, by many experts, through a series of breakthroughs, that the Rivest, Shamir and Adleman (RSA) algorithm might be broken in the next few years [Tom13]. It is suggested that RSA cryptography mechanisms should be replaced by other public-key cryptosystems and that the key sizes should be increased, which is not desirable, given the already large size of the RSA keys. This led to a renewed interest in elliptic curve cryptography, both in the industry and academia. This cryptography theory is seen as a good candidate for replacing RSA, due to its cryptographic strength and smaller key sizes [Cho] [GPW+04] [Ror] [Sta]. As emphasized below, during the development of this master's program, it was noticed that there was no readily available library implementing elliptic curve cryptography for Windows Phone 8.1, which was then prepared during one of the stages of this work.

The motivation behind this master's program was to create a security application for mobile devices taking the previously mentioned topics into account, regarding the engineering and implementation considering several functionalities and security requisites. The design of the application requires a detailed analysis of the security requirements, including a way to manage a public key infrastructure.

At the time of writing this dissertation, there was no other similar application for the specific OS for which it was implemented, which further stresses out the urge to address this absence from the Windows Phone Store.

Under the 2012 version of the Association for Computing Machinery (ACM) Computing Classification System, a *de facto* standard for computer science, the scope of the master's program reflected in this dissertation, falls within the categories named:

- **Security and privacy~Key management**

- **Security and privacy∼Public key (asymmetric) techniques**
- **Security and privacy∼Symmetric cryptography and hash functions**
- **Security and privacy∼Privacy-preserving protocols**
- **Security and privacy∼Software security engineering**
- *Security and privacy∼Mobile and wireless security*
- Security and privacy∼Security protocols.

1.2 Problem Statement and Objectives

The problem addressed in this dissertation concerns the issues partially discussed above. It tackles mostly the privacy and security issues found in the communication and file storage in mobile devices. The security applications found in the Windows Phone store have very limited functionalities and most of them still use the RSA, algorithm which is believed to be broken in the next few years, hence the need to create reliable applications based on different cryptographic theories. Given the advantages of elliptic curve cryptography and the small size of the keys, it seems to be the appropriate candidate for mobile applications.

Within the scope of the aforementioned main problem, it is possible to further identify smaller problems that need to be tackled too. For example, the public-key trust management system is also an issue, since it is necessary to ensure the validity of the exchanged keys. The encryption and decryption of files in a mobile device and assuring security of the communications comprise two different subsets of problems also. Lastly, since common users nowadays are required to remember dozens of passwords for all their accounts, and such ability can become a burden, given the complexity of such passwords, there is also a need to create a secure storage of passwords, so as to help the users keep track of their accounts and login information in a secure manner.

Given the aforementioned problem statement, the main objective of this master's program is to design and implement an easy to use and secure application for mobile devices that offers several cryptographic related functions and services. Additional objectives include:

- Identification of a range of cryptographic services that a user might need from an application like this and to deliver them in a simple and consistent manner;
- Usage of Elliptic Curve Cryptography (ECC) primitives in the implementation of the identified services, such as ECC Diffie Hellman and ECC digital signature algorithm.

- Conduct research on the topics regarding key distribution and trust, in particularly how public keys should be safely distributed and their origin proved.

1.3 Adopted Approach for Solving the Problem

After identifying the problem it was necessary to analyze it in detail and decide how to approach it. The adopted approach for solving the problem mentioned in the previous section can be summarized and structured into the following stages:

1. The first stage consists in contextualizing the problem at hand and study both the current and standard cryptographic systems and schemes being used as well as the ECC theory. In this stage is also necessary to research the different application stores for the different existing mobile OSs for applications that have similar features as the ones to be implemented throughout this master's program.

2. Stage two consists on studying and identifying the cryptographic mechanisms suited for this work. It includes the detailed definition of the functionalities to be implemented in the application.

3. Stage three concerns the software engineering of the application. Starting of by identifying the requirements and use cases and then creating the diagrams that illustrate the different use cases, activities, and components.

4. Stage four comprises the implementation of the application and of any libraries necessary that will contain the ECC primitives. It also includes testing of the application in both emulators and physical devices.

5. The last stage concerns the improvement of the application revisiting the software engineering phase, if necessary, and performing the appropriate changes.

The main body of this dissertation reflects also the adopted approach.

1.4 Main Contributions

This master's program is mainly motivated by the lack of applications providing security services in smartphones. Its main contribution is the delivery of a prototype of such an application for the Windows Phone 8.1 OS. To the best of our knowledge, this is the most complete security application for this OS at the time this dissertation was delivered. Another main contribution concerns the development of a key management scheme that associates contacts and public keys, which defines the trust and exchange infrastructure. This trust infrastructure is inspired on the Public Key Infrastructure (PKI) and Pretty Good Privacy (PGP) related schemes, but uses a more fine

grained hierarchy, given by more trust levels. The software engineering and implementation details of the aforementioned application are seen as secondary, but important, contributions of this work, mostly because they may aid future developers in the process of better devising secure applications. Aside from the smartphone application, a library was developed and made available to other developers in order to help in the development of applications that use eliptic curve cryptography. The source code of the developed libraries is provided as an open source project, under the MIT license [Ope15].
This work was submitted to the Open Mind ESEGUR 2014 contest and it was one of the semi-finalists in the awards ceremony.

1.5 Dissertation Overview

This main body of the dissertation is organized in six main chapters. The contents of such chapters can be summarized as follows:

- **Chapter 1 - Introduction** - Contextualizes the subject addressed in this dissertation by explaining its focus and scope as well as the problem statement and objectives. It includes the adopted approach to fulfill the objectives and the main contributions of this work. The structure of this dissertation is also briefly discussed at the end of the chapter.

- **Chapter 2 - Related Applications, Technologies and Preliminaries** - Analyzes some of the applications currently available for the different platforms and with a similar purpose of the application developed throughout this work. It then proceeds to describe the tools and technologies used in this master's program. Lastly, crypto related algorithms and schemes, which were important in the scope of this work, are discussed.

- **Chapter 3 - Software Engineering** - Analyzes the requirements and proceeds to the identification of the use cases and discussion of the diagrams that reflect the process of software development.

- **Chapter 4 - Security and Cryptographic Services** - Discusses the cryptograpic services that were integrated in the prototyped application, then moves on to describing the trust management system that was created, and ends up with a discussion on the security of the implemented application.

- **Chapter 5 - Implementation and Testing** - Elaborates on the implementation details of the library that contains the necessary ECC functions and of the mobile application for Windows Phone. It also includes details on how it was tested, taking advantage of both emulators and physical devices.

- **Chapter 6 - Conclusions and Future Work** - Presents the main conclusions of this dissertation as well as some directions regarding future work related with this work.

Chapter 2

Related Applications, Technologies and Cryptography Preliminaries

This chapter discusses the mobile OS for which this application was created, while comparing it with the current main OSs for mobile devices. The crypto-related preliminaries which are the base of this dissertation are also discussed.

2.1 Introduction

At the time of writing this dissertation, privacy and security were at the top of the concerns of both the academic and industry communities due disclosure of classified information and security breaches. This chapter addresses the current state of mobile OSs and the security applications that each platform has available. It discusses the technology that was used in the scope of this master's program. It also includes some preliminaries regarding cryptographic schemes that are used nowadays.

The structure of this chapter is as follows:

- Section 2.2 discusses the applications with similar functionalities to `ECCrypto` that were found for iOS, Android and Windows Phone mobile OSs;
- Section 2.3 discusses the technology that was used throughout this master's program;
- Section 2.4 gives an overview of the different cryptographic algorithms that were useful within the context of this work;
- Section 2.5 concludes this chapter with the main conclusions.

2.2 Crypto Related Applications for Mobile Operating Systems

In this section, the main mobile OSs will be discussed and an overview of the security applications that are available for each platform will be presented.

2.2.1 Android Security Applications

Android is a mobile OS developed by Google and it is based on the Linux kernel. The first version was released in September 23 of 2008 and at the time of writing this dissertation the latest released version was 5.1.1 ”Lollipop”, which was released in May 28, 2015. The marketing target of Android includes smartphones, tablets, TVs, cars and wearable devices. Android is currently the lead mobile OS, being installed in more than 80% [Ram15] of all mobile devices worldwide. It has the highest number of applications in the official Android Play Store, when compared with the official application stores for the remaining OSs.

Android has a large and active community of developers and enthusiasts who develop and distribute their own modified versions of the OS by using the source code of Android Open Source Project (AOSP).

According to Kaspersky Lab [Kas] most of the mobile malware that is detected, over 95%, is designed to target the Android platform. This is due to Android being the most widespread OS for new smartphones. A key factor is also the open nature of the Android OS, the ease with which apps can be created, and most of all, the wide variety of unofficial application markets in whose security and privacy control are often disregarded.

The most common threats affecting Android devices include Short Message Service (SMS) Trojan viruses, malware, advertising modules and exploits which are used to gain root access to smartphones.[Kas12]

Android Play Store has over 1.3 million applications available. When compared with the other OSs application stores, it has the highest number of applications.

Google Play Store [Goo15] was surveyed in the initial phase of this work so as to identify applications with similar functionalities to the ones to be implemented in this work. In order to do so, the search was focused on apps that allow storing and managing passwords, encrypting/decrypting SMS messages and emails, managing contacts and their keys, as well as file encryption capabilities. The search was narrowed to the apps that are somehow using ECC to provide their functionalities.

The highest rated apps that were found will be briefly described and their links to Android Play Store are also provided.

ECC SMS [woo14] is a paid app that allows the encryption of SMS using ECC. Its features include SMS Encryption; Uses a PKI for sharing public keys; Uses the Elliptic Curve Integrated Encryption Scheme (ECIES).

Silent Text - secure messages [Sil13] is an encrypted text messaging app with secure file transfer for mobile devices. It has the ability to send files up to 100 MB, and encrypted map locations. It uses Peer-to-peer key management, and the keys are stored on the device of the user and not on their servers. It uses the Elliptic Curve P-284 and Advanced Encryption Standard (AES) 256. However it does not seem to support SMS or email encryption/decryption, and it is used as an Internet messaging app.

SMSSecure [SMS15] is an open source app that allows the user to send SMS or Multimedia Messaging Service (MMS). It does not require an Internet connection to be used. It also allows the user to share multimedia files or attachments.

Secret Space Encryptor (SSE) - Universal Encryption App [Par15] is a password manager, text encryption and file encryption app. It does not use ECC but is included here because it is the most complete app and closest in functionalities to what is intended for the app in this dissertation. Functionalities include: password manager, using a master password; Text encryption; File Encryption; AES (Rijndael) 256bit, Rivest Cipher (RC)6 256bit, Serpent 256bit, Blowfish 256bit/448bit, Twofish 256bit, GOST 256bit + Threefish 1024bit (for S.S.E. Pro Version) ciphers are available

The conclusion is that there are not many security applications using ECC for Android. There are however many apps that allow secure messaging but most rely on RSA and the target is not SMS, but online messaging. The applications that were found to use ECC, have very limited functionalities and basically allow the encryption and decryption of text.

2.2.2 iOS Security Applications

iOS is a proprietary mobile OS developed by Apple Inc. that was originally released in 2007 for the iPhone and that is used on the mobile devices of the company such as the iPhone, iPad, iPod touch and second-generation Apple TV onward. Major versions of iOS are released annually and the latest release is iOS 8.4 which was released on June 30 of 2015. iOS currently comes second in what comes to market share, with over 18% of the market.Apple App Store is where the users can buy new apps for their devices and it currently has a ballpark of 1.2 million apps available. An analysis of applications with similar functionalities to the one developed along this work, and available in the Apple store, was also performed at an earlier stage. In order to do so, the search was focused on apps that allow storing and managing passwords, encrypting/decrypting SMS messages and emails, managing contacts and their keys, as well as file encryption capabilities. The search was narrowed to the apps that are somehow using ECC to provide their functionalities but that led to no results, however it is still important to analyze the apps regarding their functionalities so a few are included here.

The highest rated apps that were found will be briefly described and their links to Apple Store [App15] are also provided.

Pryvate - The Encryption App [Cri15] is a secure voice, email, video calls, messaging and picture sharing app. It uses RSA 4096, AES 256 and Diffie-Hellman (DH). The app also claims to never establish a direct trust relationship with any server or middle-ware, meaning that the communications are never stored or saved anywhere.

Secure Texting [App12] is an app that encrypts and decrypts text messages. It does so by using a plain text key and there is no reference regarding the algorithms being used.

CoverMe Private Texting Messenger [Cov15] is an app that features: Secure and Private Messaging; Encrypted phone Calls, secure SMS; Private vault to save personal contacts, messages, documents, pictures and videos; Password manager vault. CoverMe does not use ECC for encryption purposes. It claims to use RSA and AES for securing the comunications.

Seecrypt SC3 [See15] allows the user to make and receive, secure voice calls send and receive encrypted text messages and attachments between Seecrypt Mobile-enabled devices, anywhere in the world. Seecrypt SC3 uses a unique double-layer AES 256 and RC4 384 end-to-end crypto scheme, with new ephemeral session keys for each call and message, that affords all individuals and corporations extremely strong encrypted communication. It is an application for encrypted voice calls and Internet messaging, but does not support SMS or email.

CryptTalk Pro [Are15] is a secure call and instant messaging. It claims that the system is implemented in such a way that potential hackers are prevented from listening in, even if they have access to CryptTalk's system plans or the App's source code. It uses the maximum security through Best Practice cryptography: End-to-end encryption: a central server is not involved in the encryption process; Built on the most secure and widely used and audited algorithms; Media encryption: AES-256; Authentication: One Time Password (OTP) (Request for Comments (RFC)4226), Session Initiation Protocol (SIP) Digest; Key exchange: Elliptic Curve Diffie-Hellman (ECDH) with ephemeral keys signed using RSA-2048, end-to-end key exchange; Secure Real-time Transport Protocol (SRTP); Private key protection: Patented, proprietary method for top security authentication: Hash-based Message Authentication Code (HMAC)-Secure Hash Algorithm (SHA)1; Integrity protection: Signed Voice / Instant Messaging (IM) packets; Man-in-the-Middle (MITM) Protection: Diffie-Hellman key exchange messages signed using RSA-2048.

No applications were found for iOS to be using ECC, there are, however plenty of applications that offer some of the functionalities that are offered by the application to be implemented in this dissertation by often using weaker security algorithms.

2.2.3 Windows Phone Security Applications

Windows Phone is a mobile OS developed by Microsoft for smartphones, and it is the platform for which the application described in this dissertation was implemented. The OS will be analyzed in greater detail in the next section. For now, the discussion will be focused on related applications available in the Windows Store.

Since the platform of choice for the implementation of the mobile application was Windows Phone, it motivated a more detailed investigation regarding the available applications on the Windows Store. The criteria for searching related applications. The search was focused on apps that allow storing and managing passwords, encrypting/decrypting SMS messages and Emails, managing contacts and their keys, as well as file encryption capabilities. At some point, the search was focused to the applications that were using ECC to provide their functionalities. Once again that led to no results, as no applications using ECC were found. Nonetheless, it was still important to review some security applications that were found because they might offer some insight on how to make `ECCrypto` a better application.

Secure Text Pro [Lun15] is an app that features the ability to encrypt/decrypt messages and images and send them via SMS or email using a password. It allows the user to encrypt a message several times. Screenshots of the application can be seen in figures 2.1 and 2.2.

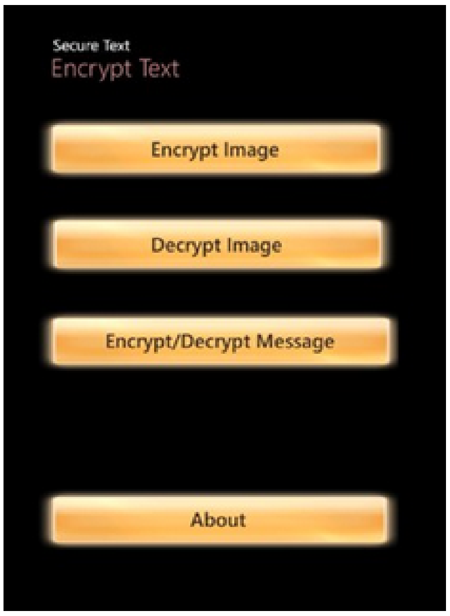
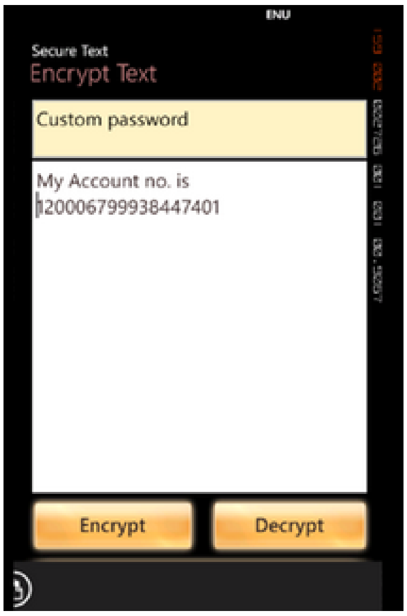


Figure 2.1: Screenshot of the main screen of the Windows Phone application Secure Text Pro.



My Account no. is
120006799938447401

Figure 2.2: Screenshot of the encryption page of the Windows Phone application Secure Text Pro.

Password Jinni [Anc15] is an app used to store private data on the phone. It stores passwords, bank accounts and credit cards information, membership data, work credentials, private contacts, web logins. It uses 256 bit AES encryption and it includes a password generator. Screenshots of the application can be seen in figures 2.3 and 2.4.

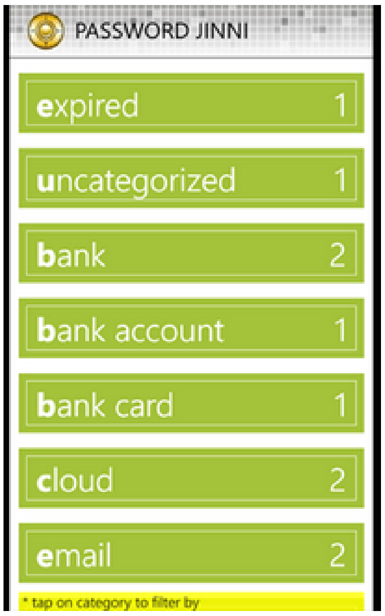


Figure 2.3: Screenshot of the main screen of the Windows Phone application Password Jinni.

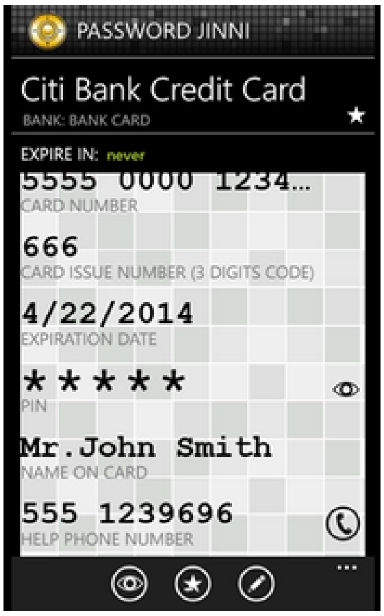


Figure 2.4: Screenshot of the credit card details screen of the Windows Phone application Password Jinni.

TXTCrypt [LFr15] is an app used to store private data on the phone. It stores passwords, bank account, credit cards information, membership data, work credentials, private contacts, web logins. It uses 256 bit AES encryption and it includes a password generator. Screen-shots of the application can be seen in figures 2.5 and 2.6.

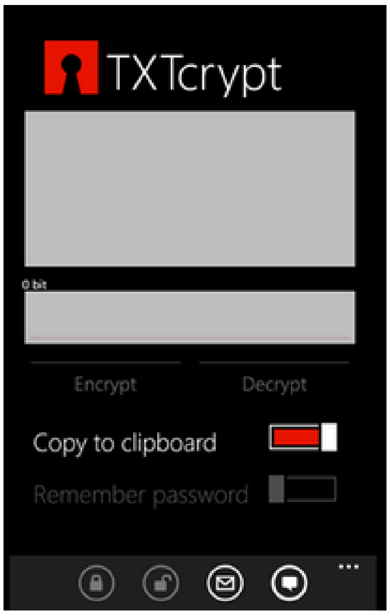


Figure 2.5: Screenshot of the encryption screen of the Windows Phone application TXTCrypt.

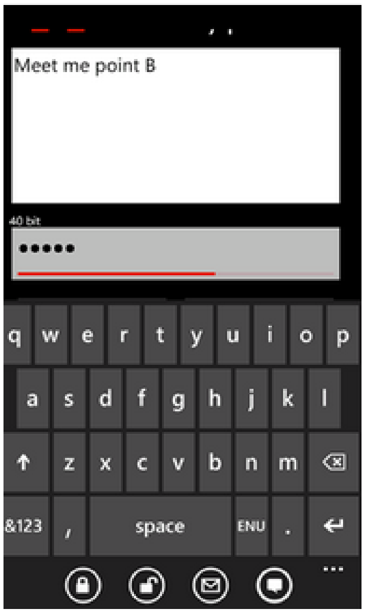
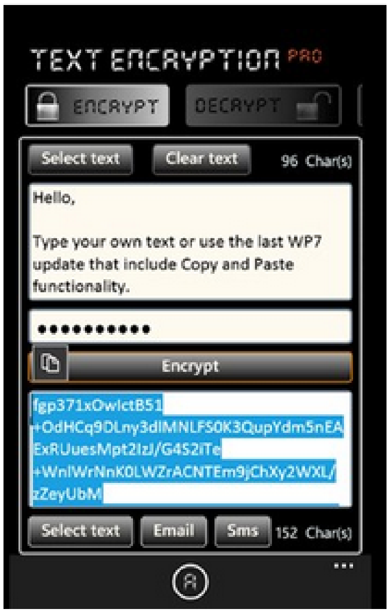


Figure 2.6: Screenshot of the decryption screen of the Windows Phone application TXTCrypt.

Text Encryption Pro [PLR15] is a text encryption application for text, SMS, and emails. It uses passwords to encrypt the messages and provides a storage mechanism to manage the passwords. Screenshots of the application can be seen in figures 2.7 and 2.8.

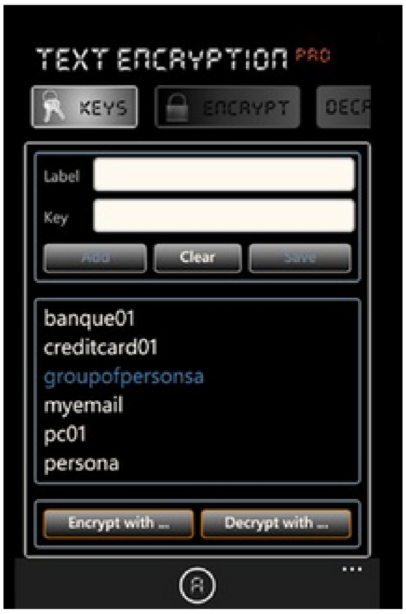


Hello,

Type your own text or use the last WP7 update that include Copy and Paste functionality.

fgp371xOwlctB51
+OdHCq9DLny3dIMNLFSOK3QupYdm5nEA
ExRUuesMpt2IzJ/G4S2iTe
+WnlWrNnK0LWZrACNTEm9jChXy2WXL/
zZeyUbM

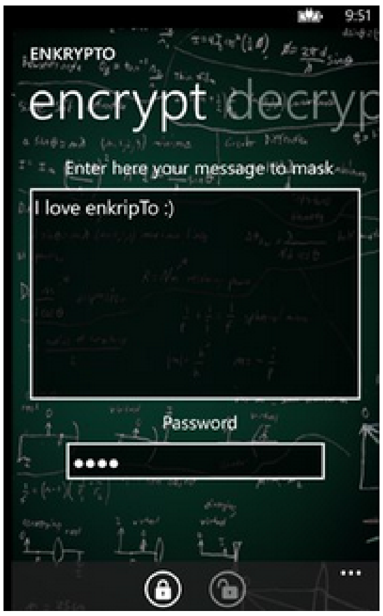
Figure 2.7: Screenshot of the encryption screen of the Windows Phone application Text Encryption Pro.



banque01
creditcard01
groupofpersonsa
myemail
pc01
persona

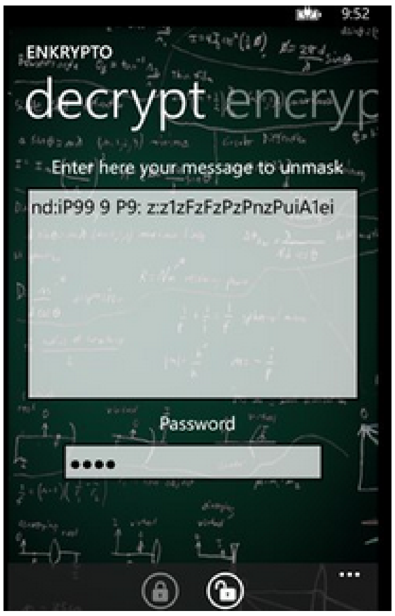
Figure 2.8: Screenshot of the key manager screen of the Windows Phone application Text Encryption Pro.

EnkrypTo [Ron15] is an encryption application able to send encrypted emails and SMS using a shared password with the message recipient. There is no information available about the algorithms being used. Screenshots of the application can be seen in figures 2.7 and 2.8.



I love enkripTo :)

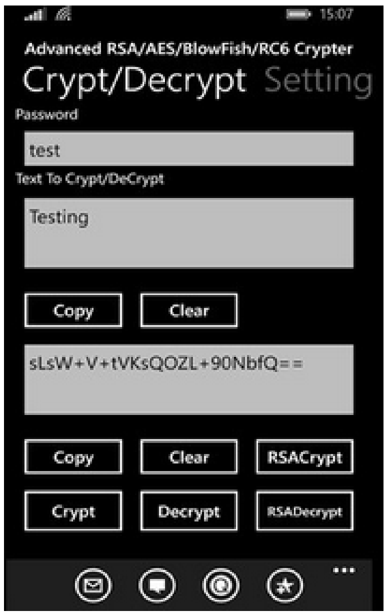
Figure 2.9: Screenshot of the encryption screen of the Windows Phone application EnkrypTo.



nd:iP99 9 P9: z:z1zFzFzPzPnzPuiA1ei

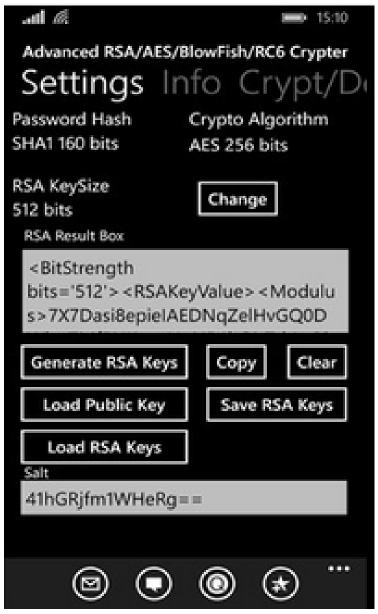
Figure 2.10: Screenshot of the decryption screen of the Windows Phone application EnkrypTo.

Crypter [Cyb15] is an encryption application with the ability to encrypt emails and SMS with AES/BlowFish/TwoFish/RC6/Serpent/Cast256 with up to 448bit symmetric keys. Secret passwords can be hashed with up to BLAKE/SHA 512bit hash. It also supports up to 8192bit public RSA key for public key exchange. It supports both symmetric and asymmetric keys. Screenshots of the application can be seen in figures 2.7 and 2.8.



Password
test
Text To Crypt/DeCrypt
Testing
sLsW+V+tVKsQOZL+90NbfQ==

Figure 2.11: Screenshot of the encryption/ decryption screen of the Windows Phone application Crypter.



Password Hash
SHA1 160 bits
Crypto Algorithm
AES 256 bits
RSA KeySize
512 bits
RSA Result Box
<BitStrength bits='512'><RSAKeyValue><Modulus>7X7Dasi8epielAEDNqZelHvGQ0D
Salt
41hGRjfm1WHeRg==

Figure 2.12: Screenshot of the settings screen of the Windows Phone application Crypter.

No applications using ECC were found for Windows Phone. A few applications offering some of the cryptographic functionalities were found, but they provide basic encryption and decryption functions only.

2.2.4 Research Conclusions

After exploring the application stores for the different platforms, it was clear that there is not much of an offer of applications with the functionalities offered by the application implemented during this dissertation. While in the Android and iOS application stores one may find applications with functionalities such as encryption/decryption of SMS and emails, password and key management, these do not use ECC in their implementation. As for the applications found in the Windows App Store, most of them simply allowed the encryption and decryption of text using a plain text password and none of them used ECC in any way. In a lot of the reviews for the applications that were found, the users often complained that the apps need more features or that the encryption methods needed to be stronger. The research also provided a better idea on how to create an app that is both pleasing and easy to use for the user as the design in such applications often seems to be disregarded.

2.3 Technology used in the Scope of this Masters

The current section will discuss the different tools and technologies that were used throughout this Masters.

2.3.1 Windows Phone 8.1

The target platform for the development of the proposed mechanism was Windows Phone 8.1. Windows Phone 8 was released on October 29, 2012 and is the third generation of the Windows Phone mobile OS from Microsoft. Windows Phone 8 was build to replace its predecessor Windows Phone 7 based in Windows CE, using the new architecture Windows NT which is also used in Windows 8 for both tablets, laptops and desktops. It is worth mentioning that Windows Phone 7 and Windows 8 architectures are not compatible and applications developed for Windows Phone 8 are not compatible with Windows Phone 7 devices.

Initially the target platform was Windows Phone 8 but on July 15th of 2014, Microsoft released Windows Phone 8.1 which is an OS update for Windows Phone 8 so it was decided to target the latest version of Windows Phone. The new version introduced various new features such as Cortana a personal virtual assistant which is similar to Siri which is the personal assistant from Apple on the iPhone and Google Now for Android. Some of the features of Cortana include the ability to recognize natural voice without the user having to input a predefined series of commands and answer questions using information from Bing search engine (such as weather and traffic informations, sport scores), the ability to set alarms and reminders, add appointments to the user's calendar, start applications and call or text contacts, among many others.

It also introduced changes to the applications framework in order to be possible to use the same application model for store apps for Windows 8.1 and Windows Phone 8.1. This was done in order to facilitate the development of applications that can be used in both phones and tablets and computers. These changes allow the developers to build "universal apps" that run on both Windows Phone 8.1 and Windows 8 and share a big portion of the code, except for the code that is specific to the platform, such as user interface and phone APIs.

Windows Phone 8.1 also included features such as support for Virtual Private Network (VPN), Bluetooth 4.0 Long-Term Evolution (LTE), allows apps to access the device's file system which was not possible in Windows Phone 8, user interface improvements, battery life improvements, a new Word Flow keyboard which allows users to swipe through letters to type, similarly to the Swype keyboard option available on Android devices.

The decision to implement the application for this platform was due to the fact that Windows Phone 8.1 is a relatively new platform and does not have many applications in the Store and none was found with the features proposed in this work.

2.3.2 Microsoft Visual Studio

For the development of the application described in this dissertation the Integrated Development Environment (IDE) of choice was Visual Studio 2013 and Visual Studio 2015 from Microsoft. Visual Studio is a comprehensive collection of tools and services that allows developers to create a wide variety of applications, from desktop applications to Windows Store applications for both tablets and Windows Phones. It can also be used to create web sites, web applications, web services, and many other projects.

Visual Studio includes a code editor that supports IntelliSense, code refactoring and the integrated debugger works has a source-level debugger as well as a machine-level debugger. It includes designers for both GUI applications, Web applications and websites, for class representation and database schema. There is also the ability to install extensions that enhance the functionalities at many levels.

Built-in languages in Visual Studio include C, C++, C++/ Common Language Infrastructure (CLI)(via Visual C++), VB.NET (via Visual Basic .NET, C# (via Visual C#), F#, JavaScript. Support for languages such as Python and Ruby among others is also available. eXtensible Markup Language (XML)/eXtensible Stylesheet Language for Transformation (XSLT), HyperText Markup Language (HTML)/eXtensible HyperText Markup Language (XHTML), Cascading Style Sheets (CSS) and JavaScript are also supported.

The programming language used in the development of this application was C# and eXtensible Application Markup Language (XAML) was used to design the visual appearance of the User Interface (UI).

Blend for Visual Studio is a visual tool used to build engaging user interfaces and rich media experiences for Windows desktop, Web, Windows Phone and Windows Store apps. It comes with the Visual Studio installation. It allows the developer to make UI changes visually in design view. Blend stores the work in XAML which can also be edited in Visual Studio. It also supports HTML5/CSS3/JavaScript for web apps and includes SketchFlow for prototyping in the Ultimate and Premium editions of Visual Studio. Blend provides a workspace that can be used to build new UI or manipulate an existing UI more quickly and intuitively. The tools available in Blend are far more extensive and sophisticated than Visual Studio's pallete which allows, for instance, the creation of animations in applications creating richer and more user-friendly applications. For the application developed in the context of this dissertation, Blend for Visual Studio 2015 was used in order to design the UI of the application.

Windows Phone 8.1 emulators are provided and added to an existing installation of Visual Studio in order to test applications without the need to have a physical device. Windows Phone

8.1 Emulators require an installation of Visual Studio with Update 2 or later. A machine with Windows 8.1 (x64) Professional edition or higher and a processor that supports Client Hyper-V and Second Level Address Translation (SLAT).

2.3.3 Microsoft Visio 2013

Microsoft Visio 2013 is part of the Microsoft Office suite and is a diagramming and vector graphics application. It allows the creation of professional diagrams to simplify complex information with updated shapes, collaboration tools and data-linked diagrams. It was used to create the necessary diagrams in this dissertation such as: database diagrams, use cases, class diagrams, and other diagrams.

2.3.4 SQLite for Windows Phone 8.1

SQLite is a library that implements a self-contained. serverless, zero-configuration, transaction SQL database engine. The library used in particular contains an extension SDK and all other components needed to use SQLite for Windows Phone 8.1 application development with Visual Studio 2013.

SQLite supports Language Integrated Query (LINQ) which is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages.

2.4 Preliminaries

In the following subsections, important crypto related algorithms and schemes in the scope of this work are discussed. The description for the elliptic curve related schemes will be more detailed in a later chapter but an overview can be found here as they constitute the most important tools of this work.

2.4.1 RSA

RSA is a scheme for public-key encryption proposed by Ronald Rivest, Adi Shamir and Leonard Adleman at the Massachusetts Institute of Technology (MIT), in 1977. Since then, RSA has became the most widely used asymmetric cryptographic scheme. RSA has many applications, but it is most often used for:

- Encryption of small pieces of data, particularly for key transport;
- Digital signatures and digital certificates on the Internet.

The RSA scheme is a block cipher in which both the plaintext and ciphertext are integers between 0 and $n-1$ for some n. Typically, n is a number with more than 1024 bits. Given a plaintext block

M and ciphertext block C, encryption and decryption are performed in the following manner:

$$C = M^e \ mod \ n$$

$$M = C^d \ mod \ n = (M^e)^d \ mod \ n = M^{ed} \ mod \ n$$

The values of n and e must be known by both the sender and the receiver. However, only the receiver knows the value of d. This is a public-key encryption algorithm with a public key $PU = \{e, n\}$ and a private key $PR = \{d, n\}$.

Certain requirements must be met in order for this algorithm to be satisfactory, such as:

- It is possible to find values of e, d, n such that $M^{ed} \ mod \ n = M$ for all $M < n$.
- It is relatively easy to calculate $M^e \ mod \ n$ and $C^d mod \ n$ for all values $M < n$.
- It is unfeasible to determine d given e and n.

Thr basic form of the RSA cryptosystem is composed of three algorithms: key generation, encryption and decryption. Assuming that two participants, Alice and Bob, wish to communicate with each other in a secure way using RSA, the three algorithms will now be explained. Key generation is required before the application of the public-key cryptosystem. The steps to generate a key are the following:

1. Select large p, q, where p and q are both prime and $p \neq q$;
2. Calculate $n = p \times q$;
3. Calculate $\phi(n) = (p-1)(q-1)$;
4. Select integer e, where $gcd(\phi(n), e) = 1 : 1 < e < \phi(n)$;
5. Calculate d, where $d = e^{-1}(mod \ \phi(n))$;
6. Public key $PU = e, n$;
7. Private key $PR = d, n$.

Once the keys are generated, if Bob wants to send a message $M < n$ to Alice, he uses the public key that belongs to Alice, and the ciphertext would be obtained in the following manner:

Ciphertext C, where $C = M^e \ mod \ n$.

As for the decryption of the message, once Alice receives the ciphertext C, she can proceed with the decryption using her private key as seen here:

Ciphertext C,
Plaintext M, where $M = C^d \ mod \ n$

2.4.2 RSA Security

The RSA algorithm can be attacked by using four possible approaches which are the following:

Mathematical Attacks aim to solve the factoring problem, which is key to the security of the algorithm. Three approaches to attacking RSA are:

1. Factor n into its two prime factors. This enables the calculation of $\phi(n) = (p-1) \times (q-1)$, which in turn enables the determination of $d = e^{-1}(mod \ \phi(n))$;

2. Determine $\phi(n)$ directly, without first determining p and q. Again , this enables the determinations of $d = e^{-1}(mod \ \phi(n))$;

3. Determine d directly, without first determining $\phi(n)$.

Brute Force Attacks are possible since RSA is as vulnerable as other cryptosystems to this kind of attack. The defense against this type of attack is to use a large key space, meaning that the larger the number of bits in d, the safer it is. It is not, however, feasible to keep increasing the number of bits in d given how complex the calculations for both the key generation and the encryption/decryption are. Thus, the larger the key size, the slower the system will run.

Timing Attacks would be the equivalent of a burglar trying the guess the combination of a safe by observing how long it takes for someone the turn the dial from number to number, The attack can be explained using the modular exponentiation algorithm used for computing $a^b \ mod \ n$. In this algorithm, modular exponentiation is accomplished bit by bit, with one modular multiplication performed at each iteration and an additional modular multiplication performed for each 1 bit. Assuming that the target system uses a modular multiplication function that is very fast in almost all cases but in a few cases takes much more time than an entire average modular exponentiation. The attack starts with the leftmost bit, b_k, and then proceeds bit-by-bit. Suppose that the first j bits are known. For a given ciphertext, the attacker can complete the first j iterations of the for loop. The subsequent step depends on the unknown exponent bit. If the bit a is set, $d \leftarrow (d \times a) \ mod \ n$ will be executed. For a few values of a and d, the modular multiplication will be slow, and the attacker knows which these are. Therefore, if the observed time to execute the decryption algorithm is always slow when this particular iteration is slow, then this bit is assumed to be 1. On the other hand, if a number of execution times for the entire algorithm are fast, then this bit is assumed to be 0. In reality, implementations of the modular exponentiation do not have such extreme timing variations, in which the execution time of a single iteration can exceed the mean execution time of the entire algorithm. Nonetheless, there is still enough timing variation making it a practical attack [Pau].

There are simple countermeasures against timing attacks, including the following:

- **Constant exponentiation time** comprises a simple fix, but it degrades performance. It works by ensuring that all exponentiations take the same amount of time before returning a result;

- **Random delay** consists in adding a random delay to the exponentiation algorithm to confuse the timing attack could improve performance. Nonetheless, Kocher [Pau] points out that attackers may still succeed by collectiong additional measurements to compensate for random delays, if the defenders do not add enough noise;

- **Blinding** can be achieved by multiplying the ciphertext by a random number before performing exponentiation. It prevents the attacker from knowing what ciphertext bits are being processed inside the computer, therefore preventing the bit-by-bit analysis, which is essential too the timing attack.

Chosen Ciphertext Attack (CCA) is an attack in which the attacker chooses a number of ciphertexts and the corresponding plaintexts, decrypted with the private key of the target by the target himself (acting as an oracle). This way, the attacker could select a plaintext, encrypt it with the public key of the target, and then be able to get the plaintext back by having it decrypted with the private key. Although this does not provide the attacker with new information, the attacker may exploit properties of the RSA and selects blocks of data that, when processed using the private key of the target, yield information needed for cryptanalysis. A simple example of a CCA takes advantage of the following RSA property:

$$E(PU, M_1) \times E(PU, M_2) = E(PU, [M_1 \times M_2])$$

$C = M^e \; mod \; n$ can be decrypted via a CCA as follows.

1. Compute $X = ((C \times 2^e) \; mod \; n$;

2. Submit X as a chosen ciphertext and receive back $Y = X^d \; mod \; n$.
   But note that

$$\begin{aligned} X &= (C \; mod \; n) \times (2^e \; mod \; n) \\ &= (M^e \; mod \; n) \times (2^e \; mod \; n) \\ &= (2M)^e \; mod \; n \end{aligned}$$

Therefore, $Y = (2M) \; mod \; n$. From this, M can be deduced.

This attack can be prevented in pratical RSA-based cryptosystems by randomly padding the plaintext prior to encryption. There are, however, more sophisticated CCAs for which a simple padding with a random value have been shown to be insufficient to provide the desired security.

2.4.3 Advanced Encryption Standard

1999 was the year the United States National Institute of Standards and Technology (NIST) indicated that the Data Encryption Standard (DES) should only be used for legacy systems and that Triple Data Encryption Standard (3DES) should be used instead. Although 3DES still resists to brute force attacks nowadays, it has several problems. It is not very efficient and its relatively short block size of 64 bits can also be a drawback in certain applications, e.g., building a hash function from a block cipher. Another concern is regarding the possibility of attacks using quantum computers, which in a few decades may become reality and for which the desirable key lengths are in the order of 256 bit. Considering all the problems of 3DES, it was concluded that a new block cipher was necessary as a replacement for DES.
In 1997, NIST called for proposals for a new AES. The following requirements for AES candidates were mandatory:

- Three key lengths were to be supported, namely 128, 192 and 256 bit;
- Block cipher with 128 bit lock size;
- Efficiency in software and hardware;
- Security relative to other submitted algorithms.

In 2001, the block cipher *Rijndael* was declared as the new AES and was published as a final standard. AES is expected to be the dominant symmetric-key algorithm for many commercial applications for the next few decades. In what regards security, to date, there are no attacks better than brute-force known against AES. AES is the mandatory encryption algorithm for United States government applications, and is also present in several open standards such as Internet Protocol Security (IPsec) and Transport Layer Security (TLS), Wi-Fi encryption standard Institute of Electrical and Electronics Engineers (IEEE) 802.11i, Secure Shell (SSH), and many others [oSN].

It is also mandatory in several industry standards and it is used in many commercial systems.

2.4.4 Elliptic Curve Cryptography

Elliptic Curve Cryptography is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. This approach was proposed independently in the eighties by two separate authors, Neal Koblitz and Victor S. Miller. An article named *Elliptic curve cryptosystems* [Kob] was published in 1987 by Koblitz; and Miller published the article *Use of elliptic curves in cryptography* in 1985 [Vic]. Despite this, only in 2004 to 2005 did Elliptic curve cryptography algorithms became mainstream only in 2004 to 2005.

Nowadays, many protocols and security applications use implementations of digital signature and Diffie Hellman key agreement protocols using elliptic curve cryptography. This approach presents very interesting advantages over other public-key cryptography mechanisms. This fact, coupled with advances in factoring composite numbers, over the last few years, make this a great alternative to the RSA public-key encryption mechanisms.

An elliptic curve over the set of real numbers, is defined by the equation:

$$y^2 = x^3 + ax + b, \text{ where } a, b \in \mathbb{R} \text{ and } -16(4a^3 + 27b^2) \neq 0$$

This definition is not yet the definition used in cryptography, though it serves the purpose of introducing the subject. The representation in the cartesian space can be observed in figure 2.13. The equation on the left represents the function $y^2 = x^3 - 3x + 3$ and its determinant is $-16(4 \times (-3)^3 + 27 \times 3^2) = -2160 < 0$. As for the representation on the right, it represents the curve given by $y^2 = x^3 - 2x + 1$ and its determinant is $-16(4 \times (-2)^3 + 27 \times 1) = 80 > 0$. If the determinant, which is given by $-16(4a^3 + 27b^2)$, is positive, then the curve has two components. Otherwise, it will only have one.

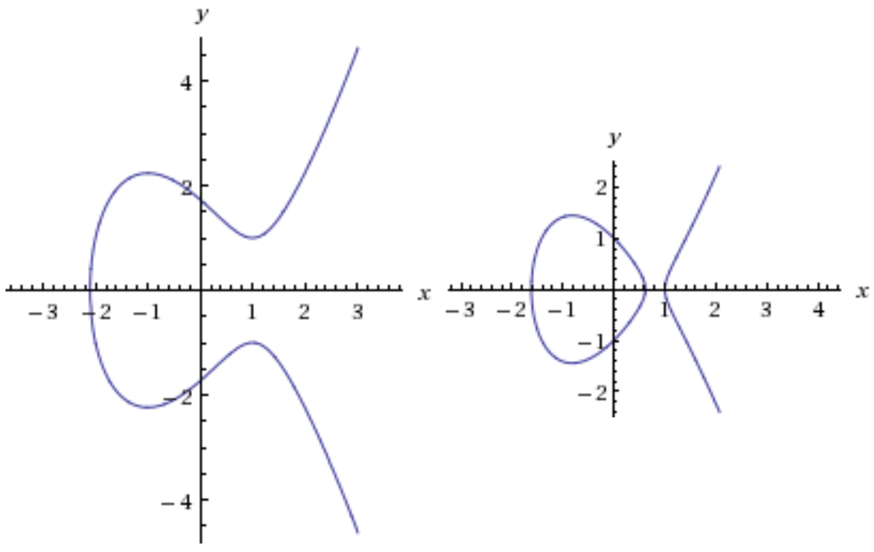


Figure 2.13: The representation of two elliptic curves on $\mathbb{R}^2$. The curve on the left represents the equation $y^2 = x^3 - 3x + 3$. As for the curve on the right, it represents the equation $y^2 = x^3 - 2x + 1$.

Public key mechanisms are typically developed following one of two possible approaches:

1. using one way functions with homomorphic properties;

2. using one way functions with homomorphic properties and with trapdoor.

One way functions mean that there is an efficient algorithm to calculate the result of the function, but not to invert it. RSA uses a function of the second type. The operation $c = m^{pk}\ mod\ P$ is a one way operation. In order to invert it, the discrete algorithm problem would have to be solved. As for the DH key exchange protocol or the ElGamal cipher, these are completely defined using one way functions without a trapdoor. In fact, ECC is based in the same type of functions and even in the same problem.

2.4.4.1 Discrete Logarithm Problem

The discrete logarithm problem [JOP14] will be explained in order to clarify why it can be used to construct crypto systems.

Given a finite cyclic group *G* and a generator *g* for that group,

$$G = 1, g, g^2, g^3, ..., g^{q-1},\ q \text{ is the order of the group,}$$

the Discrete Logarithm Problem (DAP) is hard if for all the efficient algorithms $\mathcal{A}$ defined to solve the problem, the probability to find x given g and g^x is insignificant:

$$\mathbb{P}_{g \leftarrow G, x \leftarrow Z_q}[\mathcal{A}(G, q, g, g^x) = x] < \frac{1}{2^{80}}.$$

As can be seen, the more general definition of the problem uses only cyclic groups. Therefore it can be applied to other groups different from $\mathbb{Z}_p^*$, including groups defined over elliptic curves modulo p.

There are several algorithms that try to solve the DAP [BGJT14] [Wha]. The best algorithm known to solve the DAP in $\mathbb{Z}_p^*$ is known as the General Number Field Sieve [num93], which determines the size these numbers should have in order to ensure the safety of the mechanisms that produce them. Table 2.1 summarizes and compares the requirements in terms of the size of the modulus, indicating the associated security:

As observed in the comparative table, the ratio between the safety and the size of the keys on elliptic curves is much higher than the one displayed by algorithms in $\mathbb{Z}_p^*$ groups, which include the classic RSA and Diffie-Hellman. In order to have a security of 80 bits (equivalent to 2^{80} attempts to break the mechanism), RSA keys need to have 1024 bits, while in elliptic curve it

Symmetric Scheme (key size in bits)	ECC-Based Scheme (size of n in bits)	RSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Table 2.1: Comparison between high-quality symmetric-key encryption schemes, ECC and RSA in terms of key strength.

would only require keys to have 160 bits. In general, the security offered by ECC keys is half its size in bits. The management, transmission and storage of ECC keys is therefore simpler than the management, transmission and storage of RSA keys.

2.4.4.2 Definition of Elliptic Curves

ECC elaborates on the concept of cyclic group but defined for elliptic curves. In Algebra, a cyclic group is a set of elements that can be generated from one (or more) of its elements, called generator g, from the application of the group operation repeatedly to that generator. Each element can then be written as a power of g if the multiplicative notation is used, or as a multiple of g if the additive notation is used. The group $\mathbb{Z}_7^*$, for example, can be generated by the element 3:

$$\{3^1, 3^2, 3^3, 3^4, 3^5, 3^6\} = \{3, 2, 6, 4, 5, 1\},$$

where the group operation $\times$ is defined by $a \times b \ mod \ 7$, in which $a, b \in \mathbb{Z}_7^*$.

The elliptic curve over $\mathbb{Z}_p,\ p > 3$, is the set of all pairs $(x, y) \in \mathbb{Z}_p$ fulfilling the equation

$$y^2 = x^3 + ax + b \ mod \ p,$$

together with an imaginary point of infinity $\mathcal{O}$, where $a, b \in \mathbb{Z}_p$ and $4a^3 + 27b^2 \neq 0 \ mod \ p$. According to the definition of an elliptic curve, it is required that the curve is nonsingular. From a geometrically point of view, the plot must have no self-intersections or vertices. This can be achieved if the discriminant of the curve $-16(4a^3 + 27b^2)$ is nonzero.

The presented definition is different than the one that was presented initially. Its graphical representation is also very different from the one included before, as it is shown in figure 2.14.

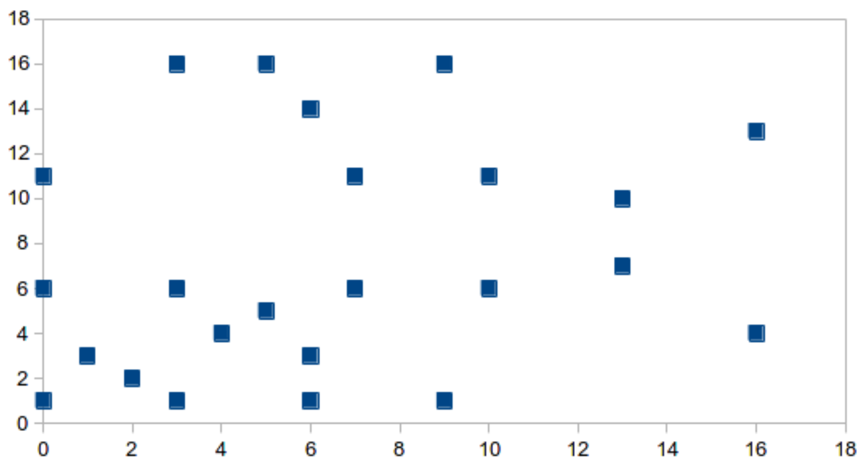


Figure 2.14: Representation of the elliptic curve $y^2 = x^3 + 2x + 2 \; mod \; 17$.

2.4.4.3 Group Operations on Elliptic Curves

In order to obtain a cyclic group with the desired properties, ECC inventors defined also the addition of points in the curve. Although it is usually represented by $+$, this operation should not be confused with the addition of real numbers. The definition of addition in elliptic curves is done with two points of the curve, A and B, represented by coordinates (x_1, y_1) and (x_2, y_2). As a result of the addition of these two points, a new point in the curve is obtained, with the coordinates (x_3, y_3), respecting the condition $y^2 = x^3 + ax + b \; mod \; p$.

The representation of points of the curve $y^2 = x^3 + ax + b \; mod \; p$ is not *human friendly*, but representations of curves in the cartesian plane can be used for a geometric interpretation of how points can be summed in elliptic curves. Figure 2.15 shows the sum of points A and B. It also illustrates where the identity ($\mathcal{O}$) is *located*. This point does not seem to be on the curve (although one side of the curve tends to $+\infty$ when $x \rightarrow +\infty$).

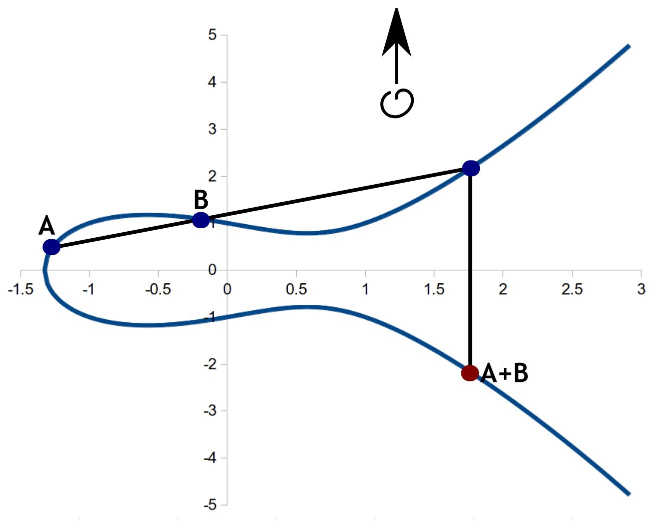


Figure 2.15: Addition of point A and point B according to the definition of sum in elliptic curves.

In the image it can be observed that the sum of two points is interpreted as the symmetric point (xx axis) to the one obtained after the interception of the curve with the line that passes through A and B. It can be concluded that, given the curve configuration and the definition of this operation, the resulting point is always another point of the curve. Although calculating the intersection point for $\mathbb{R}^2$ curves is simple, for curves with the modulus operation it is not so straightforward.

There are some particular cases that need to be addressed. For instance, the addition of a point with itself, meaning the multiplication of said point by two. The addition of point C with itself is illustrated in figure 2.16. It is defined that the sum is given by the symmetric point to the one that is obtained after the curve is intercepted with the tangent line to the point.

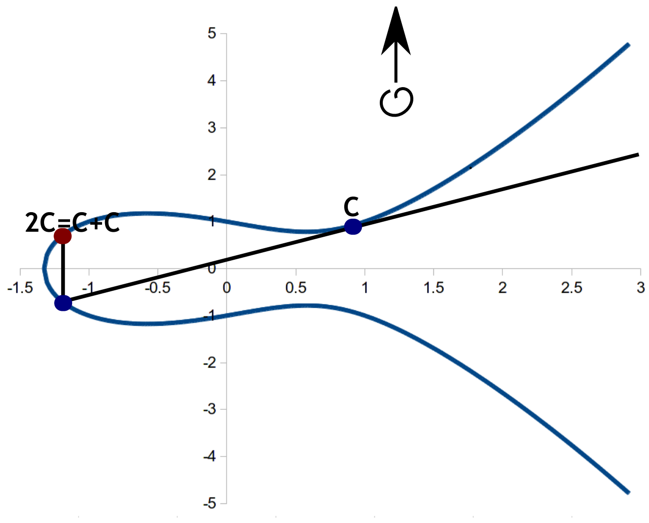


Figure 2.16: Addition of point C with itself.

Both cases that were discussed cover the possible situations, except for the sum with the identify. It is possible to resort to the following formulas to calculate point addition and multiplication by 2 in cryptographic elliptic curves. Consider that $A = (x_1, y_1)$ and $B = (x_2, y_2)$. The coordinates of $A + B = (x_3, y_3)$ are respectively calculated by:

$$x_3 = s^2 - x_1 - x_2 \text{ mod } p \text{ and}$$
$$y_3 = s(x_1 - x_3) - y_1 \text{ mod } p,$$
$$\text{in which}$$
$$s = \begin{cases} (y_2 - y_1) \times (x_2 - x_1)^{-1} \text{ mod } p, & \text{if } A \neq B \\ (3x_1^2 + a) \times (2y_1)^{-1} \text{ mod } p, & \text{if } A = B \end{cases}.$$

It should be noted that the formulas include the multiplication for an inverse modulo p, which should not be treated as an integer multiplication.

The multiplication of a point by a scalar is the most important operation in ECC. It is in this operation that the discrete logarithm problem is defined for ECC, as it can be observed next. Considering an elliptic curve E, parameterized in order to be cryptographically secure, a point P and a point T, the discrete logarithm problem consists in finding the integer number d such that:

$$dP = P + P + P + ... = T$$

For carefully chosen curves, the multiplication of a point P by a scalar appears to produce a random result in the curve and it is a function that is hard to invert. Thus, it is a one way function that can be used in a Diffie-Hellman protocol. The fast modular multiplication is a fast and efficient algorithm for calculating the multiplication of a point for a scalar. The multiplication of a point C by different scalars is illustrated in figure 2.17.

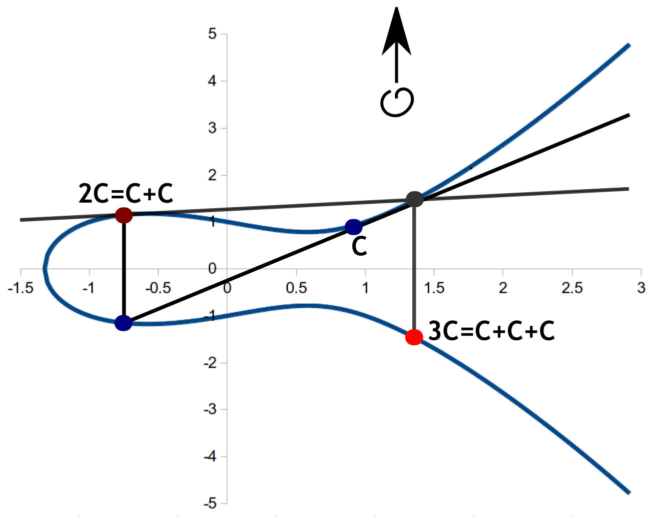


Figure 2.17: Multiplication of point C by 2 and 3.

Lastly, the sum of a point with the identify is a special case. The choice of this point, as well as the definition of the addition, was performed in a way that the sum of a point with the identify is itself. When a point D is summed with $\mathcal{O}$, one must draw a line originating from $+\infty$ and that passes by D, intersecting the curve in the symmetric point (according to the xx axis) to D. Since the addition operation uses the symmetric point to this intersection, the resulting point of is D, $D + \mathcal{O} = D$). Figure 2.18 illustrates the addition of a point in cryptographic elliptic curves with the identify.

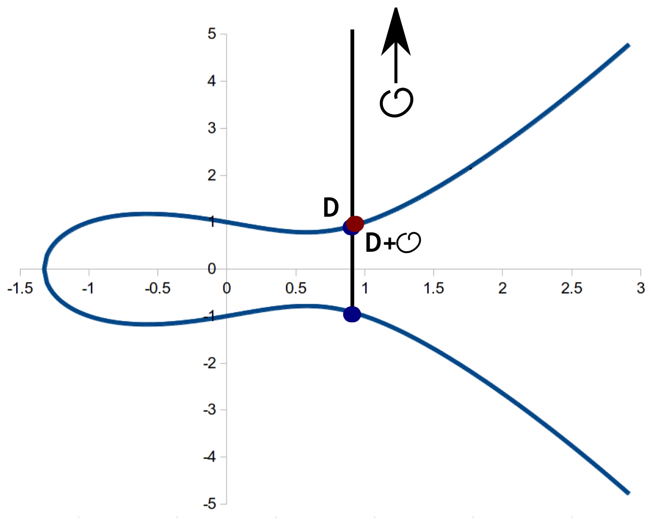


Figure 2.18: Addition of point D with $\mathcal{O}$ in cryptographic elliptic curves.

The generation of the domain parameters is time-consuming and troublesome to implement, therefore, several standard curves and domain parameters have been established(American National Standards Institute (ANSI)1, Federal Information Processing Standards (FIPS),Standards for Efficient Cryptography (SEC)2, Brainpool[M. 10])

The United States NIST has set some recommended elliptic curves for government use and is currently the standard being used for federal use. National Security Agency (NSA) Suite B Cryptography [Nat09] is a set of cryptographic algorithms promulgated by the NSA as part of its Cryptographic Modernization Program and it serves as an interoperable cryptographic base for both unclassified information and most classified information. The components of Suite B are: AES, Elliptic Curve Digital Signature Algorithm (ECDSA), ECDH. SHA2 and RSA.

NIST Recommended Curves

According to the SEC by Certicom Corp., certain curves and domain parameters are recommended for ECC [Res]. The Elliptic Curve (EC) domain parameters over $\mathbb{F}_p$ are the sextuple,

$$T = (p, a, b, G, n, h),$$

where p is an integer specifying the finite field $\mathbb{F}_p$, $a, b \in \mathbb{F}_p$ are two elements specifying an elliptic curve $E(F_p)$ defined by the equation:

$$E : y^2 = x^3 + ax + b \ (mod\ p).$$

$G(x_G, y_G) \in E(\mathbb{F}p)$ is termed Base Point, n is a large prime number which is the order of G, and an integer h which is the cofactor $h = \#E(\mathbb{F}_p)/n$.

According to the Standards for Efficient Cryptography [Res], elliptic curve domain parameters over $\mathbb{F}_p$, must fulfill the condition

$$[log_2 p] \in 112, 128, 160, 192, 224, 256, 384, 521.$$

Such restriction is implemented in order to encourage interoperability and allow implementers to supply commonly required security levels. Elliptic curve domain parameters over $\mathbb{F}_p$ with $[\log_2 p] = 2t$ provide approximately t bits of security. Hence, it is believed that approximately 2^t operations are necessary for solving the discrete logarithm problem on the associated elliptic curve.

Two different types of elliptic curve domain parameters over $\mathbb{F}_p$ are mentioned in the Standards for Efficient Cryptography [Res].

Parameters associated with a Koblitz curve give rise to particularly efficient implementations. Klobitz curves are binary anomalous curves over $\mathbb{F}_2^m$ which have $a, b \in 0, 1$ [Kob]. The Klobitz curve recommended parameters were chosen by repeatedly selecting parameters admitting an efficiently computable endomorphism until a prime order curve was found.

Verifiably random parameters are chosen from a seed using SHA-1. This process ensures that the parameters cannot be predetermined. Therefore, it is highly unlikely that the parameters can be susceptible to future special-purpose attacks, and during their generation no trapdoors can have been placed in the parameters.

The recommended elliptic curve domain parameters over $\mathbb{F}_p$ are represented in the following manner:

1. Each name begins with sec to denote *Standards for Efficient Cryptography*;
2. Followed by a p to denote parameters over $\mathbb{F}_p$;
3. Number denoting the length in bits of the field size p;
4. Followed by a k if the parameters are associated with a Koblitz, or an r if the parameters are verifiably random;
5. Lastly, a sequence number is added.

Table 2.2 summarizes the recommended elliptic curve domain parameters over $\mathbb{F}_p$. The column labeled *Strength* gives the approximate number of bits of security that the parameters offer. The column *Size* gives the length ()in bits) of the field order. The column labeled *RSA/DSA* gives the approximate size of an RSA or Digital Signature Algorithm (DSA) modulus at comparable strength. The last column indicates whether the parameters are associated with a Koblitz curve (k) or were chosen verifiably at random (r).

Parameters	Strength	Size	RSA/DSA	Koblitz or random)
secp112r1	56	112	512	r
secp112r2	56	112	512	r
secp128r1	64	128	704	r
secp128r2	64	128	704	r
secp160k1	80	160	1024	k
secp160r1	80	160	1024	r
secp160r2	80	160	1024	r
secp192k1	96	192	1536	k
secp192r1	96	192	1536	r
secp224k1	112	224	2048	k
secp224r1	112	224	2048	r
secp256k1	128	256	3072	k
secp256r1	128	256	3072	r
secp384r1	192	384	7680	r
secp521r1	256	521	15360	r

Table 2.2: Comparison between the NIST recommended curves, ECC and RSA in terms of key strength.

Another standard used nowadays concerns the so-called Brainpool curves. The reason for which NIST curves were chosen instead was due to their better performance when compared with Brainpool Curves [Pau13].

The performance of ECDSA and of ECDH is greatly impacted by the choice of the elliptic curve being used. The differences in performance will now be illustrated. When comparing the information in tables 2.3 and 2.4, the performance of the NIST recommended curves is much better than the Brainpool curves for ECDSA for both signing and verifying a signature.

NIST Curve	Relative Speed
ECDSA-secp521r1	549 sign/s
ECDSA-secp384r1	859 sign/s
ECDSA-secp256r1	1148 sign/s
ECDSA-secp224r1	1707 sign/s
ECDSA-secp192r1	2190 sign/s
ECDSA-secp521r1	151 verify/s
ECDSA-secp384r1	233 verify/s
ECDSA-secp256r1	333 verify/s
ECDSA-secp224r1	491 verify/s
ECDSA-secp192r1	670 verify/s

Table 2.3: NIST curves performance for ECDSA.

Brainpool Curve	Relative Speed
ECDSA-brainpoolP512r1	65 sign/s
ECDSA-brainpoolP384r1	126 sign/s
ECDSA-brainpoolP256r1	203 sign/s
ECDSA-brainpoolP512r1	15 verify/s
ECDSA-brainpoolP384r1	28 verify/s
ECDSA-brainpoolP256r1	52 verify/s

Table 2.4: Brainpool curves performance for ECDSA.

As for performance regarding ECDH, NIST recommended curves are once again much better than the Brainpool curves as seen in tables 2.5 and 2.6.

NIST Curve	Relative Speed
ECDH-secp521r1	157 handshake/s
ECDH-secp384r1	248 handshake/s
ECDH-secp256r1	334 handshake/s
ECDH-secp224r1	511 handshake/s
ECDH-secp192r1	716 handshake/s

Table 2.5: NIST curves performance for ECDH.

Brainpool Curve	Relative Speed
ECDH-brainpoolP512r1	15 handshake/s
ECDH-brainpoolP384r1	28 handshake/s
ECDH-brainpoolP256r1	52 handshake/s

Table 2.6: Brainpool curves performance for ECDH.

2.4.5 Elliptic Curve Diffie-Hellman

The ECDH is a variant of the Diffie-Hellman protocol using elliptic curve cryptography. It is a key agreement protocol that is anonymous and allows two parties to establish a shared secret over an insecure channel. The shared secret can be used as a key or used to derive one. Using a symmetric key cipher, it is then possible to encrypt subsequent communications. The ECDH protocol works as follows:

1. Alice (A) and Bob (B) agree to use an elliptic curve that is considered secure with known and public parameters (*p, a, b, G, n, h*). *p* is a prime number, *a* and *b* define the curve, *G* is the generator point in the curve, *n* is the order of the group and *h* is the co-factor (the ratio between the number of points in the curve and *n*);

2. A: $d_A \leftarrow^r 1, ..., n-1, X = (x_1, y_1) = d_A * G$;
   Alice generates a random number between 1 and $n-1$ and calculates $X = (x_1, y_1) = d_A * G$, i.e., the sum of G, d_A times. d_A is a secret value and X is a public value.

3. A $\rightarrow$ B:X;
   Alice sends Bob her public X value.

4. B: $d_B \leftarrow^r 1, ..., n-1, Y = (x_2, y_2) = d_B * G$;
   Bob generates a random number between 1 and $n-1$ and calculates $Y = (x_2, y_2) = d_B * G$, i.e., the sum of G, d_B times. d_b is a secret value and Y is a public value.

5. B $\rightarrow$ A:Y;
   Bob sends Alice his public Y value.

6. A: $K = (x_3.y_3) = d_A * Y = d_A * (d_B * G)$;
   Alice calculates the shared key from the public Y value that belongs to Bob and her own secret value d_A.

7. B: $K = (x_3.y_3) = d_B * Y = d_B * (d_A * G) = d_A * (d_B * G)$;
   Bob calculates the shared key from the public X value that belongs to Alice and his own secret value d_B.

A scheme of the ECDH protocol can be seen in figure 2.19.

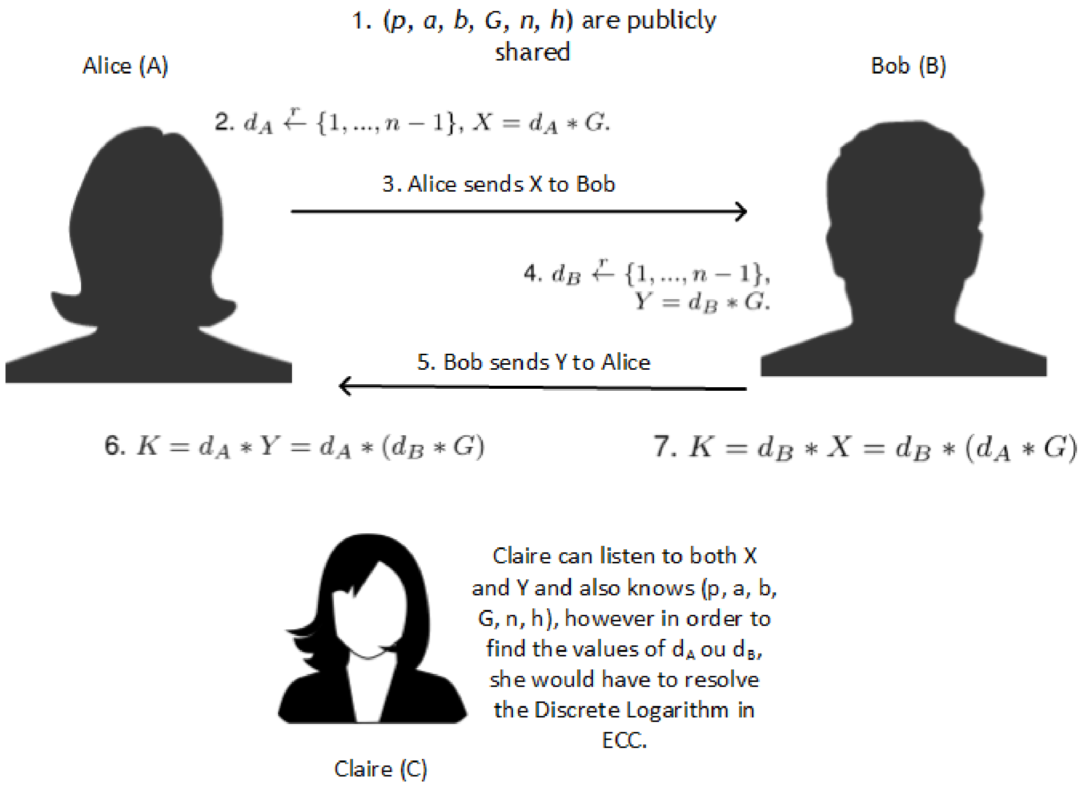

1. (p, a, b, G, n, h) are publicly shared
Alice (A)
Bob (B)
2. $d_A \xleftarrow{r} \{1, ..., n-1\}$, $X = d_A * G$.
3. Alice sends X to Bob
4. $d_B \xleftarrow{r} \{1, ..., n-1\}$, $Y = d_B * G$.
5. Bob sends Y to Alice
6. $K = d_A * Y = d_A * (d_B * G)$
7. $K = d_B * X = d_B * (d_A * G)$
Claire can listen to both X and Y and also knows (p, a, b, G, n, h), however in order to find the values of d_A ou d_B, she would have to resolve the Discrete Logarithm in ECC.
Claire (C)

Figure 2.19: Elliptic Curve Diffie-Hellman Key agreement protocol.

2.4.6 Elliptic Curve Integrated Encryption Scheme and Digital Signature

The *Integrated Encryption Scheme* is a hybrid encryption system, that combines both symmetric key and public key cryptography mechanisms. If implemented correctly, it will ensure the semantic safety in plaintext and chosen cyphertext attack models. When it is applied to elliptic curves, the scheme is called ECIES.

The scheme assumes the usage of a symmetric encryption algorithm, represented by the algorithms (E, D), and defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$; of a secure Key Derivation Function (KDF), defined from $\mathbb{Z}p$ to K^2, and of a Message Authentication Code (MAC), defined for $(\mathcal{K}, \mathcal{C}, \mathcal{T})$. $\mathcal{K}$ denotes the key space, $\mathcal{M}$ denotes the message space, $\mathcal{C}$ denotes the cryptogram space and $\mathcal{T}$ denotes the space of all possible MACs. It is also assumed that an elliptic curve with parameters that are considered safe (p, a, b, G, n, h) is chosen. The three algorithms of ECIES can now be defined.

- The public and private keys generator F is specified in the following manner:

  1. Choose a random number d_A in $1, ..., n-1$ and calculate $X = (x_1, y_1) = d_A * G$;

  2. Return $sk = d_a$ and $pk = (X, (p, a, b, G, n, h))$.

- The encryption algorithm $E(pk,,m)$ works in the following way:

    1. Choose a random number $d_B \in 1, ..., n-1$ and calculate $Y = (x_2, y_2) = d_B * G$;
    2. Calculate $K = (x_3, y_3) = d_B * X$. The secret S between both is now x_3;
    3. Use the KDF to derive two keys k_1 and k_2, one to encrypt, and the other to calculate the MAC;
    4. Encrypt the message with k_1, i.e., $c \leftarrow E(k_1, m)$;
    5. Calculate the MAC of the cryptogram with k_2, i.e., $t \leftarrow MAC(k_2, c)$;
    6. Concatenate and return Y, the cryptogram and the MAC, i.e., (Y, c, t).

- The decryption algorithm $D(sk, (Y, c, t))$ is as follows:

    1. Derive the common point $K = (x_3, y_3) = d_B * Y$. The secret S between both is then x_3;
    2. Use a KDF function to derive two keys k_1 and k_2, one to decrypt, and the other one to verify the MAC;
    3. Verify MAC and return failure if $t' \neq MAC(k_2, c)$;
    4. Decrypt m from c using k_1, i.e., $m = D(k_1, c)$.
    5. Return m.

The fact that a cryptographic derivation function of secure keys is used in order to generate symmetric encryption keys as well as the MAC, together with the discrete logarithm problem, is enough to prove the safety against Chosen Plaintext Attack (CPA).

This crypto system is often called hybrid because it uses a symmetric-key cipher to encrypt the data (e.g., AES), and an asymmetric cryptography technique to exchange the key used in the symmetric cipher. Since an ephemeral symmetric key is generated for each message, it guarantees security, but there is a cost associated: the size of the cryptogram is always larger than that of the plaintext because it is necessary to send the cryptogram as well as the public value Y (which is used in the decryption). The fact that the same plaintext can lead to multiple cryptograms (depending on the encryption key), makes of this cipher a probabilistic cipher.

The ECDSA was standardized in 1998 in the United States by ANSI due to the advantages it has over RSA or over other schemes such as Elgamal and DSA. The shorter bit length of ECC keys results in shorter processing times and in shorter signatures.

2.5 Conclusion

This chapter discussed the research regarding the cryptographic applications available for the different mobile OSs. Any of the three available platforms could have been chosen to develop the application described herein since all of them lack such a complete application. However, Windows Phone seemed to be the most appealing one, due the lack of security applications in its app store and because of how recent it is, when compared with the other two platforms.

It is also clear that the security of current cryptographic algorithms may not last much longer and that a new, stronger, standard must be implemented, ECC compromises a good candidate for the job. The small size of its keys, together with the difficulty of solving the Discrete Algorithm Problem using ECC makes it a fine choice in terms of performance and security.

Chapter 3

Software Engineering

3.1 Introduction

This chapter discusses the software engineering of the application developed along this master's program. It addresses the requirements analysis that was done prior to the development phase. Unified Modeling Language (UML) diagrams are used to describe the uses cases, and activities. Database diagrams can also be found in this chapter.

The structure of this chapter is as follows:

- Section 3.2 enumerates the requirements;
- Section 3.3 describes the most important identified use cases;
- Section 3.4 elaborates on how the database was designed;
- Section 3.5 discusses the activity, class and components diagrams that are part of the software engineering process;
- Section 3.6 contains the main conclusions of this chapter.

3.2 Requirements Analysis

In this section, the identified requirements will be explained. Requirements can be categorized in multiple ways and, in this dissertation, the approach will be to focus on the functional and non-functional requirements, which will be analyzed in the following subsections.

3.2.1 Functional Requirements

Functional requirements identify necessary tasks, actions or activities that must be accomplished. The `ECCrypto` application should allow:

- Generation of password and cryptographic secrets with basis on sources of true randomness available in mobile devices;
- Generation of elliptic curve key pairs (for encryption, for digital signature, etc.);

- Establishment of secret keys between two or more smartphones users using ECDH;
- Encryption of written messages, with the possibility to export the ciphertext via email, SMS or other available sharing options;
- Decryption of written messages;
- Digital signature of files with the ECDSA;
- Importing, exporting and handling of public keys, as well as all the functions required to support the trust infrastructure;
- Authentication via public-key cryptography;
- Encoding and decoding of messages;
- Secure storage of files in the device;
- Management of passwords.

3.2.2 Non-functional Requirements

Non-functional requirements specify criteria that can be used to judge the operation of a system, but not specific behaviors. Given the fact that this application targets a mobile OS, which is used in smartphones with the usual characteristics of having a small screen, on-board keyboard, etc., it is fulcrum that it provides a pleasant user experience. System and app performance is also key, so that the user takes the most advantage of the app and is not frustrated by long response times. Some of the non-functional requirements that were identified are as follows:

- The application should be intuitive and easy to understand;
- The application layout should be simplified and the different functionalities should be grouped accordingly;
- The application appearance should be clean and simple;
- The response times should be less than 1 second for normal functions, and 2.5 seconds for cryptographic functions.

3.3 Use Cases

This section presents the use cases that were identified during the preliminary analysis of the application to be implemented. Use cases are diagrams that represent the features provided by the app to the actors. As for the identified actors that interact with the `ECCrypto`, only the *User* was considered, since it is the only one that interacts with the application and takes advantage of all features.

3.3.1 General Use Case

The most general use case shows an overview of the main features that can be found in `ECCrypto` as depicted in figure 3.1. These include the management of passwords, user keys, contacts, messages and files. Notice that all main use cases include logging into the application.

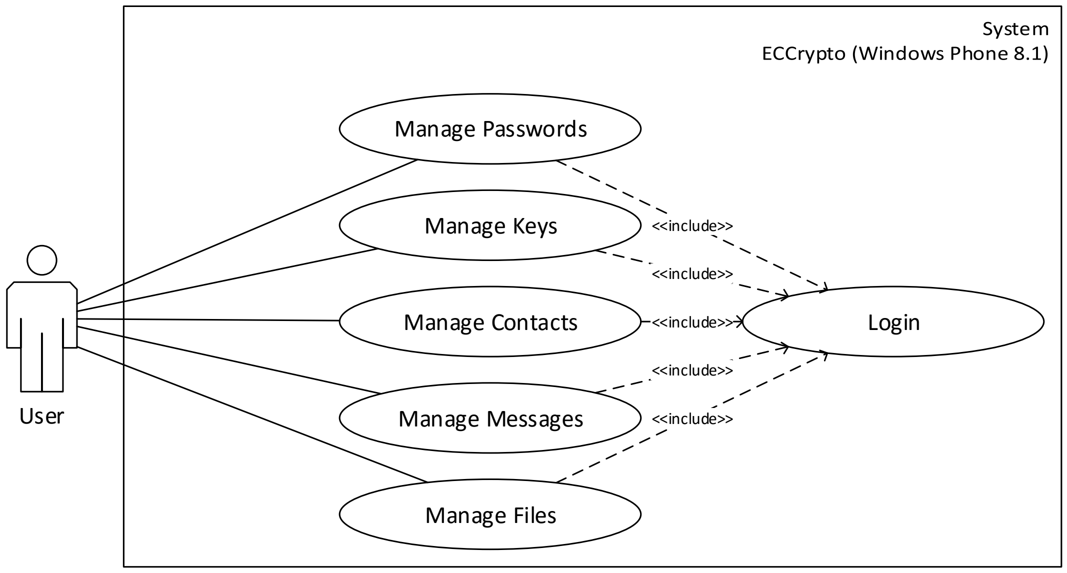


Figure 3.1: General `ECCrypto` use case diagram.

3.3.2 Detailed Use Cases

The use case previously shown in figure 3.1 will now be explained in detail by analyzing the use cases that are part of the overall use case.

Starting by the use case diagram shown in figure 3.2, one can observe the possible different actions that a user has when interacting with the application pertaining to the *password management* system. As seen in the diagram, the user can add a new password, view or edit a previously stored one, and can also delete a password. When adding or editing a password, the user has the choice of either generating a password automatically using the password generator provided by the application or enter a password manually.

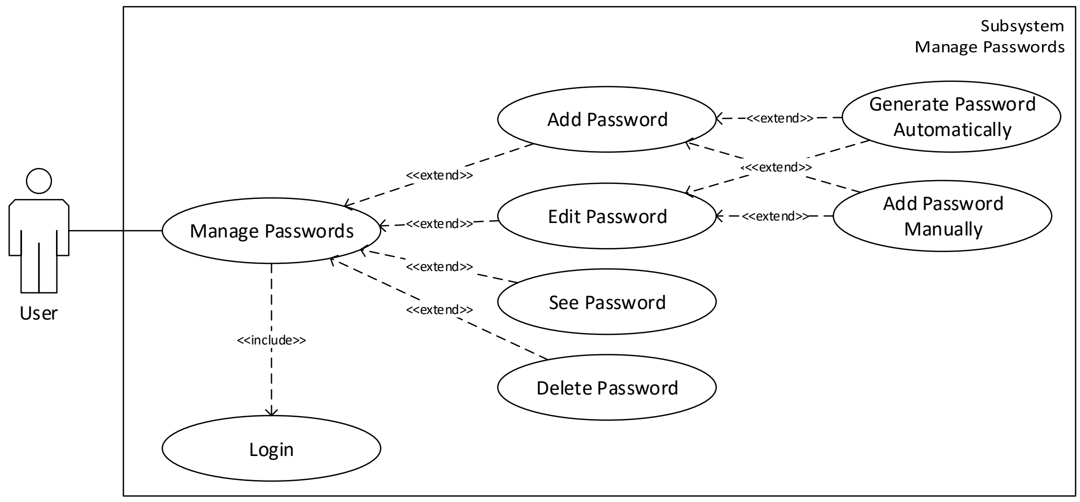


Figure 3.2: Use case diagram for the *Manage Passwords* subsystem.

In figure 3.3, the use case diagram shows the different actions that the user can perform when interacting with the user *key management* system of the application. As seen in the diagram, the user can generate a new key, edit and delete a user key that was previously stored.

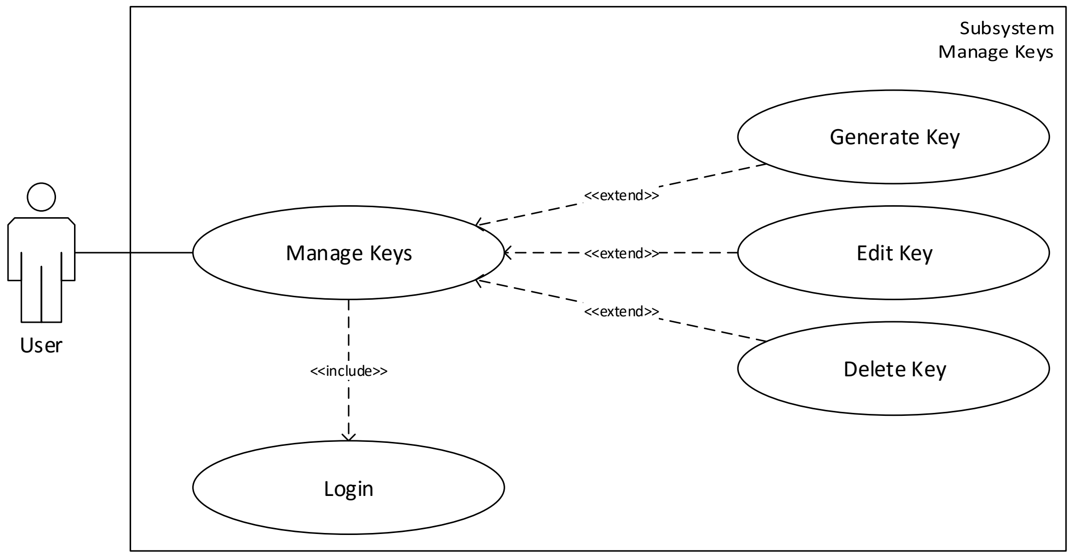


Figure 3.3: Use case diagram for the *Manage Keys* subsystem.

The actions that the user has when interacting with the *contacts management* system can be seen in the use case depicted in figure 3.4. As seen in the diagram, the user can add a new contact, import a key for the contact, edit or delete a previously stored contact.

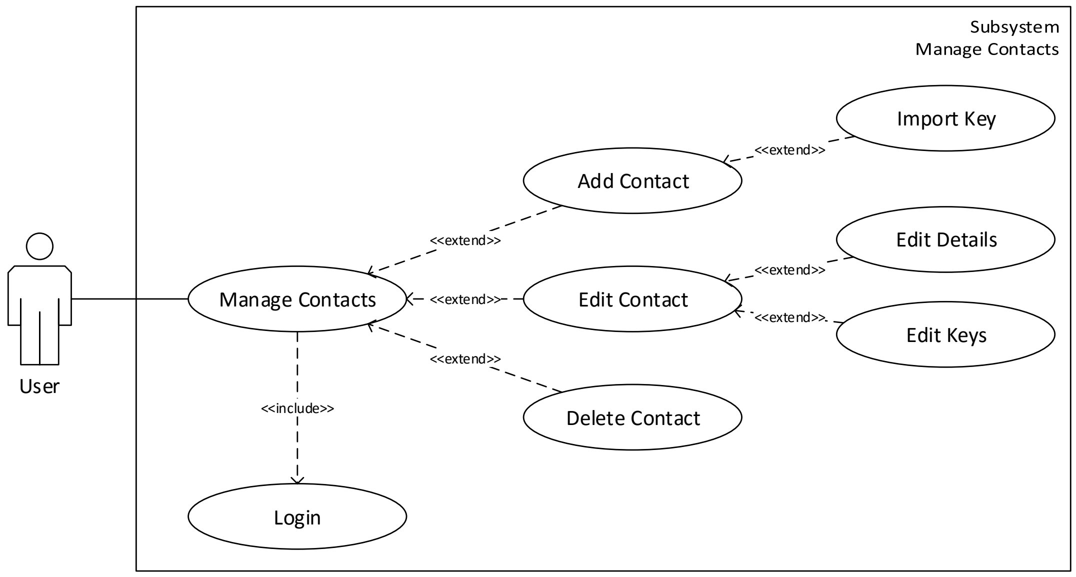


Figure 3.4: Use case diagram for the *Manage Contacts* subsystem.

In figure 3.5, the use case diagram shows the possibilities that the user has when interacting with the application in regards to the *messages management* system. As seen in the diagram, the user can encrypt and decrypt messages, as well as delete messages that were previously stored.

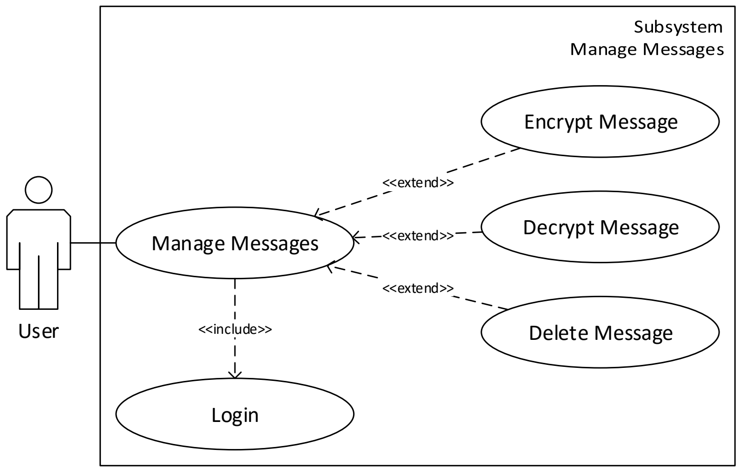


Figure 3.5: Use case diagram for the *Manage Messages* subsystem.

In figure 3.6, the use case diagram shows the actions that the user can perform in regards to the *files management* system. As seen in the diagram, the user can encrypt and decrypt files.

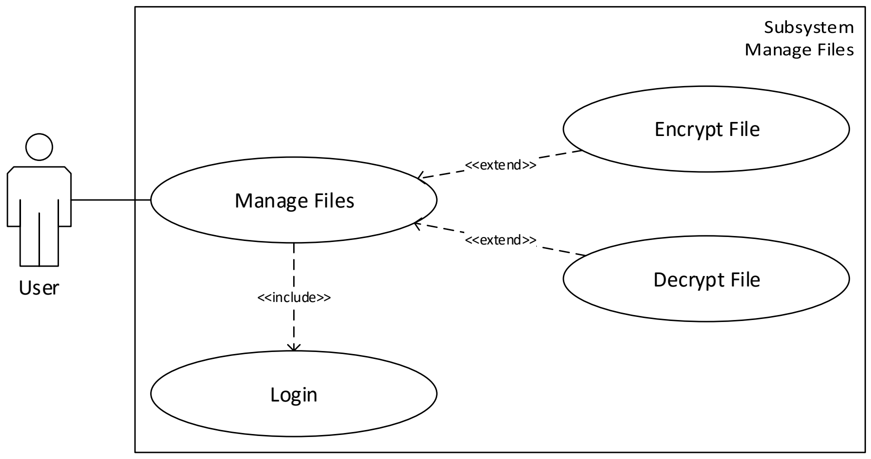


Figure 3.6: Use case diagram for the *Manage Files* subsystem.

3.4 Database Design

Given that it was necessary to store information for the application, it was decided that a local database was the best option. This section includes the diagrams depicting how the database was designed are explained.

3.4.1 Entity-Relationship

The Entity-Relationship model for the database is depicted in figure 3.7 and shows how the different entities are related to each other. A `Contact` may be related with zero or more `ContactKey` entities. On the other hand, a `ContactKey` entity may only be related to one `Contact` entity. A `ContactInfo` entity can be associated with only one `Contact`, while a `Contact` can have zero or more associations with the `ContactInfo` entity. One message can only be associated with one `ContactInfo`, however, a `ContactInfo` can have zero or more associations with the `Message` entity. The `Password` and `UserKey` entities are not related to any of the other entities.

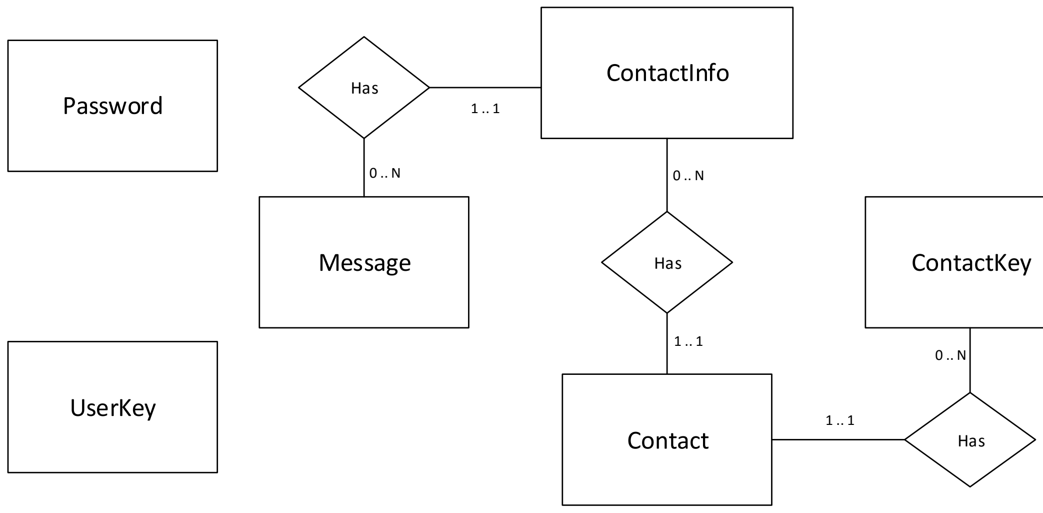


Figure 3.7: Entity-Relationship diagram of the ECCrypto database.

3.4.2 Relational Model

The Relational Model can be seen in figure 3.8, which shows in detail the tables that were created in the database, as well as the fields that every table contains.

The Password table holds the information of the stored passwords. The stored information includes a username, a password, a name that can be used to describe the password, and the date in which the password was added. A unique identifier for each password is also included.

The UserKey table holds the keys that belong to the user. It contains a name, which the user can use to easily identify the key, a longer description if desired, the public and private key of the user and lastly, the date in which the key was added. It includes the unique identifier for each UserKey.

The Contact table contains the information of the contacts that were added by the user. It stores the first and last name of the user in two separate fields and the date that the user was added. It also includes the unique identifier for each contact.

The ContactKey table contains the keys that belong to the contacts that have been added. It includes the user public key, a description that the user can add to more easily identify, the trust level of the key, the number of times the key was validated by other contacts, the date in which the key was added and the necessary Identifier (ID) to relate the data with the Contact table.

As for the ContactInfo table, it holds the different contact information of the contacts, meaning that, it includes the the phones or emails that the user can add to a contact. The field Name is available so that the user can more easily identify the contact info, by giving them names (e.g., *Work phone*, *Personal Email*). The `Info` field contains the email address or the phone number. The `IsPhone` field is a boolean that is *true* if the contact info is of the type *phone* or it is *false* if it is of the type *email*. The date in which the information was added is also stored in the table. The unique identifier for the contact info is included, as well as the ID that relates this table to the `Contact` table.

Lastly, the messages that were exchanged can be found in the `Message` table, which includes the ID of the contact info that was used in the exchange of the message, the subject of the message, and the message content. It also includes the `Sent` field, which indicates whether the message was sent or received and the date in which the message was sent or received. The unique identifier for the message is included, as well as the ID that associates the message with the `ContactInfo` table.

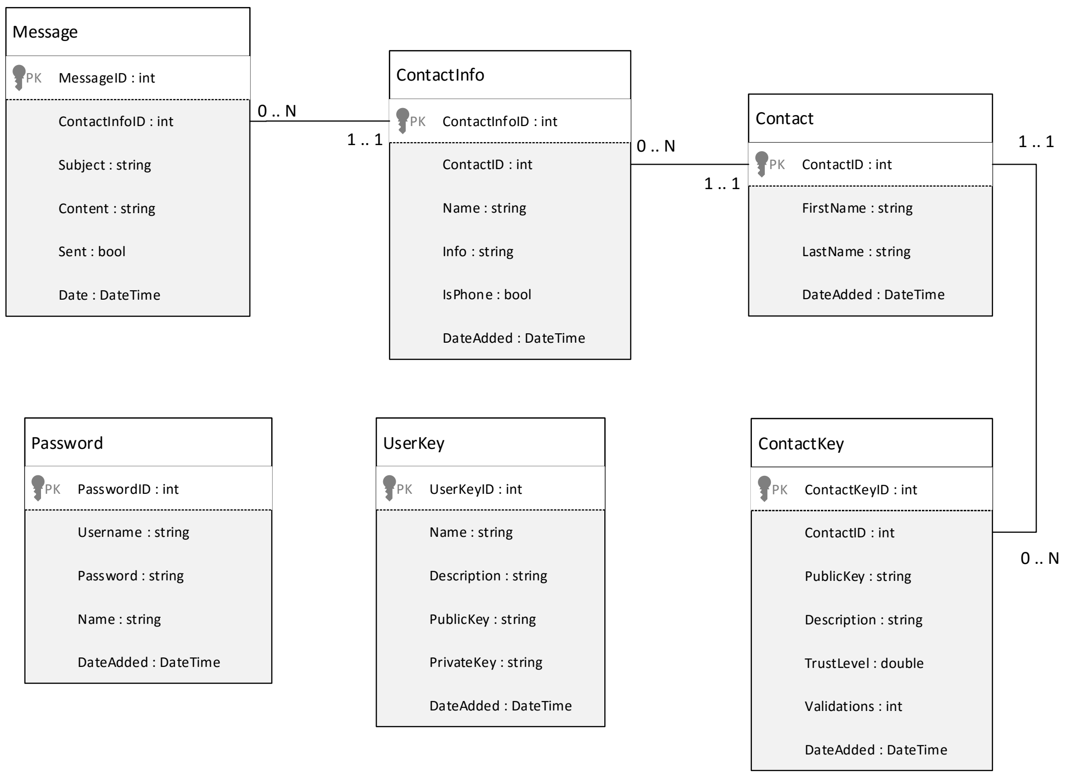


Figure 3.8: Relational Model diagram of the `ECCrypto` database.

3.5 Activity, Components and Class Diagrams

This section introduces the activity, components and class diagrams. It is divided into three main subsections, each corresponding to the appropriate diagram.

3.5.1 Activity Diagrams

Activity diagrams are used to show the overall flow of control. They are graphical representations of workflows of actions and activities in a stepwise approach which have support for choice, iteration and concurrency. In this subsection, some activity diagrams that are considered to be more representative or complex are going to be presented and briefly discussed.

The `Encrypt File` activity diagram is illustrated in figure B.2 in Appendix B. This diagram shows the steps necessary for the user to encrypt a file using the `ECCrypto` application.

The `Add Password` activity diagram is represented in figure B.1 in Appendix B. The diagram shows the steps necessary for the user to add a new password to the `ECCrypto` application.

The `Share Key` activity diagram is illustrated in figure B.3. The diagram shows the steps that a User A must take in order to share one of his personal keys with another User B using the Quick Response (QR) code generator of the application as well as the smartphone camera.

The `Import Key` activity diagram is illustrated in figure B.4. The diagram shows the opposite of the share key activity. It shows the steps necessary so that the User B is able to add the key of the User A to the contact keys using the QR code generator of the application, as well as the smartphone camera.

3.5.2 Components Diagram

The components diagram can be observed in figure 3.9. It shows how all the different components are wired together in order for `ECCrypto` to work. As seen in the image, it shows the three different main elements which are: the phone application project (`ECCrypto.exe`), the shared library project for the phone application logic (`ECCryptoLib.dll`),, which is where all the cryptographic logic of the application can be found, the `ECCryptoLibTest` that was used to unit test the library. Lastly, it also shows the external dependencies, such as libraries that were used.

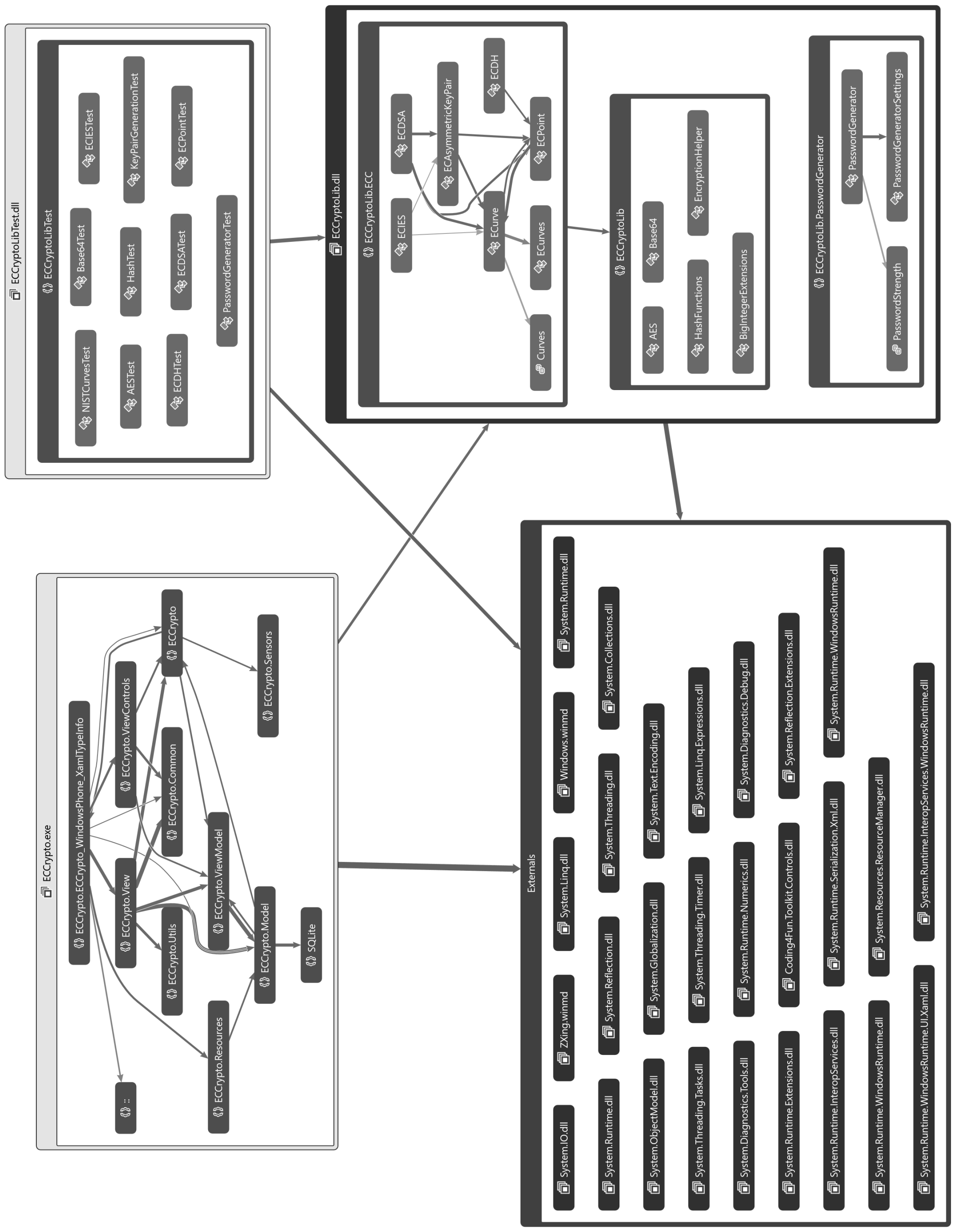


Figure 3.9: Diagram that shows all the different components of the `ECCrypto` application.

3.5.3 Class Diagram

Class diagrams illustrate the structure and relationships between classes used in the implementation of a system. Given the complexity and the size of the diagrams, only the diagram for the library project `ECCryptoLib` will be shown and explained. The diagram can be found in figure 3.10, and the several classes can be enumerated as follows.

1. `BigIntegerExtensions` includes some methods that were necessary to perform some curves calculations using the `BigInteger` class but are not provided by the .NET framework;
2. `HashFunctions` contains all the SHA and HMAC-SHA algorithms;
3. `Base64` has the Base64 conversion methods;
4. `EncryptionHelper` contains some helper methods to facilitate encryption and decryption;
5. `PasswordGenerator` is where the methods for generating and validating a password are found;
6. `PasswordGeneratorSetting`is where the different settings to generate a password are defined;
7. `PasswordStrength` stores some enums used in the password generation;
8. `ECurve` contains the structure the domain parameters of an elliptic curve;
9. `ECurves` contains the domain parameters of the implemented elliptic curves;
10. `Curves` stores the enums created to easily identify the elliptic curves;
11. `ECPoint` is where the definition of a point in an elliptic curve can be found, as well as the operations necessary;
12. `ECAsymmetricKeyPair` contains the appropriate methods to generate keypairs;

ECPoint is where the definition of a point in an elliptic curve can be found, as well as the operations necessary;

13. `ECDH` contains the methods required to perform the ECDH;
14. `ECIES` implements the methods required to perform encryption and decryption using ECIES;
15. `ECDSA` contains the methods required to digitally sign and verify signatures using ECDSA;
16. `AES` is the class where the encryption and decryption methods using AES can be found.

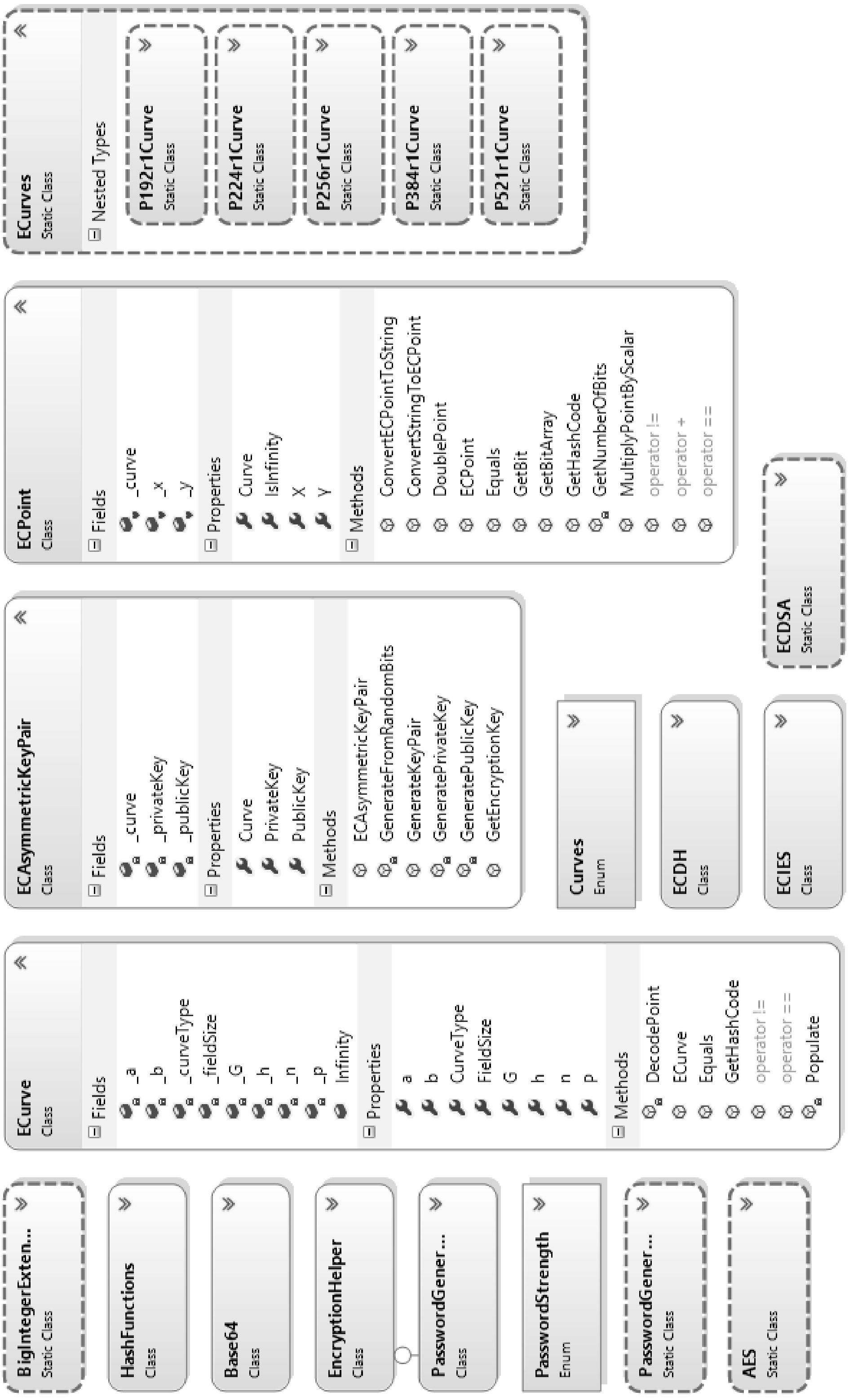


Figure 3.10: Class Diagram of the `ECCryptoLib` library.

3.6 Conclusion

This chapter focused on analyzing the requirements that were identified in the initial phase of this master's program and prior to the development of the application. It included the definition of features such as generation of user keys, encryption and decryption of messages, and management of passwords amongst others. Different diagrams that illustrate how the software engineering process took place and what to expect from the final application were presented. Use cases, activity diagrams, and components diagrams were included, as well as the diagrams that show how the database was designed. This phase provided the building blocks for the prototyping of the application, whose report is included in the following chapter.

Chapter 4

Security and Cryptographic Services

4.1 Introduction

In this chapter, the security and cryptographic services implemented in the `ECCrypto` application will be discussed. The definition of the Trust Management system will be also analyzed and, finally, there will be a discussion regarding the security of the application. The chapter is structured in the following manner:

- Section 4.2 discusses the cryptographic services identified and implemented in the `ECCrypto`;
- Section 4.3 explains the cryptographic services implemented in the `ECCryptoLib`;
- Section 4.4 describes the trust management system created in the scope of this work;
- Section 4.5 is a discussion on several security aspects regarding `ECCrypto`;
- Section 4.6 concludes this chapter with the main conclusions.

4.2 Cryptographic Services in ECCrypto

The cryptographic services that were identified to be implemented in the `ECCrypto` application will be discussed in this section.

4.2.1 Password and Cryptographic Secrets Generation with Basis on Sources of True Randomness Available in Mobile Devices

Mobile devices nowadays contain several sensors that can provide a good source of true randomness. These include sensors like accelerometers, gyroscopes, compasses, accelerometers, light sensors, etc. Using the information that these sensors provide is a great source of true randomness since it would be nearly impossible to replicate all the conditions of the phone at the same time. In order to improve the randomness generator, the different sensors that are available on the phone will be used when true sources of randomness are required in the application, such as password generation and key generation.

4.2.2 Generation of elliptic curve key pairs

A main feature of `ECCrypto` is the generation of the elliptic curve asymmetric key pairs. They are at the core of the application since they are necessary for the ECC functionalities, such as ECDH, ECDSA, encryption and decryption. The application should allow a user to generate new keys and manage them.

4.2.3 Establishment of secret keys between two or more smartphones users

The application should allow for the exchange of secrets in a secure way. In order to achieve this ECDH should be implemented and used to guarantee that the key exchanges are performed safely.

4.2.4 Encryption and Decryption of Written Messages

One of the main objectives of the application is to allow the user to communicate in a secure way. In order to do so, all the messages that are exchanged need to be encrypted by the sender and decrypted by the recipient of the message. The encrypted messages can then be sent as emails, SMS, or sent through some other channel at the choice of the user by being able to copy the encrypted text and pasting it wherever desired.

4.2.5 Digital Signature of Files with the Elliptic Curve Digital Signature Algorithm

Digital signatures provide assurance that the message or file was sent by a known sender, and the sender cannot deny having sent the message or file. Besides guaranteeing the authenticity of the message or file, it also guarantees the integrity of the message or file, i.e., that it was not altered in any way in transit. Through the use of ECDSA, `ECCrypto` should guarantee the authenticity of files and messages.

4.2.6 Importing, Exporting and Handling of Public Keys

Given that the application uses an asymmetric public key mechanism, it is necessary to manage the different keys. It includes the management of the keys of the user, as well as of the keys that belong to the contacts of the user. The application should allow the user to generate key pairs using elliptic curve cryptography also. The user should have the ability to edit and delete user keys that were previously stored. It is also required that the contacts that are added by the user can be managed, edited and deleted if necessary. An import mechanism is necessary to easily exchange keys between contacts.

4.2.7 Authentication via Public-key Cryptography

It is important to guarantee that the key exchange is performed properly and in a way that the user can be sure that a key belongs to a certain contact. This led to the definition of a trust management system that will help the user verify if a key truly belongs to a contact. This will be done resorting to exchanging a key in person using a QR code or through the trust mechanism in which a key can be verified by contacts that are common to both users.

4.2.8 Encoding and Decoding of Messages and Secure Storage of Files on the Device

Encoding of a message is the process of putting a sequence of characters (letters, numbers, punctuation and certain symbols) into a specialized format for efficient transmission or storage. Decoding is the opposite process, the conversion of an encoded format back into the original sequence of characters. The messages that are sent and received by `ECCrypto` should always be encoded in Base64 format. `ECCrypto` should permit the encryption and decryption of files on the device that in part of public folders.

4.2.9 Management of Passwords

One of the main features of `ECCrypto` is the ability to store the passwords of a user in a way that is both safe and practical. The reality is that, with all the accounts that a user has nowadays, it is extremely hard to remember all the passwords, and a common mistake is to have the same password for multiple accounts, which leads to security concerns if an account is breached, because then all the others will also be at risk.

4.3 Cryptographic Services in ECCryptoLib

In order to have the necessary cryptographic functions for the app, it made sense to create a library that would contain all the cryptographic logic that was required. In order to do so, a portable library was created that can easily be imported by the `ECCrypto` application or by other applications that require such cryptographic services. The library includes functions for the generation of passwords, calculating hash functions, generation of asymmetric key pairs using ECC, ECDH and ECDSA, among others. The different cryptographic mechanisms that were implemented will now be discussed.

4.3.1 Password Generation

As part of the Password Management system, when a new password is added or an existing one is edited, the user has the choice of either entering a password manually or let the application generate one. The `ECCryptoLib` contains a class for password generation and validation. The decision to add this functionality to the library, instead of to the `ECCrypto` application, was because this a functionality that might be useful in other scenarios and this way it is available to any other projects by simply importing the library. It allows for the generation of a random password with a different set of customizable properties. The number of characters in the password, whether it should contain numbers, non-alphanumeric characters, upper and lower case letters are all settings that can be changed in the application settings.

4.3.2 Key Pair Generation

The library permits the generation of asymmetric key pairs using elliptic curves. The keys that are generated should be in accordance with the elliptic curve that is chosen (SecP192r1, SecP224r1, SecP256r1, SecP384r1 ,SecP521r1). The generation of keys includes true sources of randomness, such as the sensors in the smartphone.

4.3.3 Elliptic Curve Diffie Hellman

ECDH is required to exchange public keys in a secure way in the `ECCrypto` application. The necessary ECDH functions are included in the library so that it can be imported by application and used.

4.3.4 Encryption and Decryption

Functions that will perform the encryption and decryption of messages and files are included in the `ECCryptoLib`, as it will be used throughout the whole application and for the most features of the application.

4.3.5 Elliptic Curve Digital Signature Algorithm and Hash Algorithms

The ECDSA offers a variant of the DSA which uses elliptic curve cryptography. `ECCrypto` should digitally sign files and messages in order to guarantee authenticity, integrity and non-repudiation. To accomplish this, the ECDSA needs to be included in the suite of algorithms that are part of the `ECCryptoLib` to then be used by the application. The necessary hashing and message authentication algorithms should be included in the `ECCryptoLib`. The algorithms include: SHA-1, HMAC and others that may be of interest to include in the library.

4.4 Trust Management

A trust management system was created to guarantee the trust in the keys of the contacts. The level of trust in a key is a percentage value and it varies depending on the number of contacts that confirm the trust in said key.

If a key is exchanged in person, via QR code exchange, it is considered that the trust in the key is 100%. However, if a key is added manually by the user, it will be considered untrusted until other contacts verify the key or if the user verifies the key in person.

The number of confirmations from other contacts required for a key to be fully trusted should be able to be altered in the settings of the application. By default, the number of required confirmations by other users is three.

4.5 Discussion on Security

Different topics regarding the security of the `ECCrypto` application as well as of the Windows Phone 8.1 devices will now be discussed.

4.5.1 Windows Phone 8.1 Security Overview

The Windows Phone OS takes advantage of the latest standards-based security hardware components to help protect devices and the information stored within them [Mic]. The main security technologies, standards and mechanisms supported by the Windows Phone OS are as follows:

Unified Extensible Firmware Interface (UEFI) is a standards-based replacement for the typical Basic Input/Output System (BIOS) that is found on most devices. It adds security features and advanced capabilities while providing the same functionalities as BIOS. UEFI is required for Trusted Boot;

Trusted Platform Module (TPM) is a tamper-resistant security processor that is designed to help secure data, ensure integrity and enable authentication;

Address Space Layout Randomization (ASLR) protects the system and applications by moving executable images into random locations within system memory. It makes it difficult for an attacker to exploit vulnerabilities that may be discovered in the platform or in applications;

Data Execution Prevention (DEP) is a defensive technology that dramatically narrows the attack surface area for memory related exploits. It prevents the execution of executable code from memory sections that were allocated for read only data;

Device encryption based on BitLocker Drive Encryption technology for data stored on the device is supported by all Windows Phone devices;

AppContainer sandbox is a sandoboxing mechanism that offers fine-grained security permissions and inherently blocks unauthorized access to the system, applications, and data. Applications only have access to capabilities that are declared in the application manifest. Capabilities that applications use are automatically published to the application details in the Windows Store. Applications are isolated from each other and can only communicate using predefined communications channels and data types;

Encryption of applications on removable storage was added as an improvement in Windows Phone 8.1. Applications can be installed on a Secure Digital (SD) card and stored on a hidden partition that is specifically designated for this purpose. The partition is encrypted just like the internal storage;

Microsoft Security Development Lifecycle (SDL) is a software development process that is used by Microsoft to ensure that development teams create software that is secure by design and can eliminate, or at least mitigate potential security risks. According to the Cisco 2014 Annual Security Report [Cis], Windows Phone environment contains far less malware than peers such as Apple iOS and Android.

Taking this for granted, it is possible to say that the application is secure. For instance, an Android developer can in a very simple manner create an application that appears to be working normally and in the background be sending spam messages either by email or SMS, if such is allowed in the application manifest. In Windows Phone, such a situation could never happen because an application cannot send an SMS or email without the knowledge of the user. Sending an email or a SMS in the Windows Phone OS, is done by calling the Compose Task in which the message is composed. The fields, such as the phone number, message content can be prepopulated by the application but the user must always press the send button for the message to be sent. Although this was initially viewed as a constraint in the development of `ECCrypto`, the security reasons behind it are solid and end up enhancing the security of the application itself. Another issue that arose throughout the development was the inability to copy text to the system clipboard. Although it can be somewhat of a nuisance to force the user to select and copy the text if it is to be sent through some channel other than email or SMS, it guarantees that applications cannot intercept the text that is being copied and pasted by the user. This is particularly important considering that the user will likely copy the passwords that are stored in the application thus increasing the security of the application.

The password that is set by the user when the application starts is stored encrypted in isolated storage using the `PasswordVault` class, which is part of the `Windows.Security` namespace. The contents of the vault are specific to the application. Apps and services do not have access to credentials associated with other applications or services. The database is stored in the isolated storage of the application, which can only be accessed by the application and it is stored encrypted.

Using ECDH to perform key exchanges ensures that two parties, each having an elliptic curve public–private key pair, can establish a shared secret over an insecure channel. This means that by using ECDH to exchange the keys, one can ensure that all the communications that follow are done in a secure way. On the other hand, ECDSA ensures that all the messages that are exchanged are signed, guaranteeing authenticity.

QR codes were used to facilitate the exchange of keys between users, and to ensure total trust in the key that was exchanged by the users. When a key is added manually, the trust level on the key will then depend on the number of users that verify that the key belongs to the contact. The number of users that are required to ensure full trust in the key can be altered in the application settings.

The first time the application is ran, the user is advised to set a numeric password to lock the application. The user is prompted to input the password every time the application is started, in order to guarantee that no one can access the information contained in the `ECCrypto` application.

4.6 Conclusion

In this chapter, the cryptographic services to be provided by both the `ECCrypto` and `ECCryptoLib` were discussed. The `ECCrypto` cryptographic services include the generation of asymmetric key pairs, encryption and decryption of messages, storage and management of passwords, file encryption and decryption, among others. The `ECCryptoLib` is to contain all the cryptographic functions that are required in the application, maximizing code portability and constituting one of the main contributions of this work. Having explained the main services offered by the library, the trust management scheme and discussed some security details, the discussion can now flow into the presentation of the prototype and some of the tests performed.

Chapter 5

Implementation and Testing

5.1 Introduction

In Visual studio, related projects can be grouped in a single solution which means that in this particular case, the following projects are part of the `ECCrypto` Solution:

- `ECCryptoLib`;
- `ECCryptoLibTest`;
- `ECCrypto`.

In this chapter, the implementation of `ECCrypto` application and of the `ECCryptoLib`, which contains the cryptographic functions necessary for the functioning of the application, will be discussed. It will also discuss how the application and library were tested. The structure of this chapter is as follows:

- Section 5.2 describes the implementation of the`ECCryptoLib`;
- Section 5.3 describes the implementation of the `ECCrypto` application;
- Section 5.4 discusses the testing of the `ECCrypto` application and of the `ECCryptoLib` library;
- Section 5.5 explains the submission process of the application to the Windows Store and of the library to GitHub;
- Section 5.6 completes this chapter with the main conclusions.

In the following subsections the different projects and their functionalities will be explained.

5.2 ECCryptoLib Implementation

As previously mentioned, a portable library was created to be used in both Windows Phone 8.1 and Windows desktop applications. The library contains the cryptographic functions that are used by the application.

The library contains the implementation of the ECC algorithms. The implementations did not resort to using external libraries for the elliptic curve functions because, at the time that the library was implemented, there was no library available targeting both platforms for which the library was created. This led to the implementation of all the functions that were required to use ECDH and ECDSA, including the methods for the different operations on elliptic curves. `ECCryptoLib` also includes the implementation of hash algorithms, a password generator, a QR code generator, the AES functions required to provide the encryption/decryption features of the application, among other helper functions such as base64 conversion methods and extensions for the BigInteger class.

5.2.1 Hash Functions

A class was created that implements several hash functions taking advantage of methods available in the `Windows.Security.Cryptography` namespace. Some of which were not used in the `ECCrypto` application, but were added as they may be of use in the future for someone using the library. The following hash functions were added to the library: SHA-1, SHA-256, SHA-384 SHA-512 and Message-Digest 5 (MD5)

HMAC functions were also necessary in the `ECCrypto` application so they were also added. HMAC is a specific construction for calculating a MAC involving a cryptographic hash function in combination with a secret key. It is used to verify the data integrity and the authentication of a message. In order to calculate an HMAC, a cryptographic hash function is used, such as SHA-1, SHA-256 or MD5. The cryptographic strength of the underlying hash function, the size of the hash output as well as the size and quality of the used key all affect the cryptographic strength of the HMAC. The output size of the HMAC is the same of the one of the hash function that was used.

Even though not all of the HMAC functions were used after implementation, the following HMAC functions were included in this class for the same purpose that the other hash functions were added: HMAC-SHA-1; HMAC-SHA-256; HMAC-SHA-384; HMAC-SHA-512; HMAC-MD5.

5.2.2 AES

Message and file encryption and decryption in `ECCrypto` use the previously mentioned AES algorithm with a key size of 256 bits. The `Windows.Security.Cryptography` namespace, contains the required methods to encrypt and decrypt both messages and files using AES, so the required functions for encryption and decryption were added to the library in the AES class.

5.2.3 Password Generator

As mentioned in the previous chapter, the library contains a password generator which allows the generation of passwords according to the specifications of the user. A class was added to the library in which the methods required to generate passwords were added. It also contains methods that are used to calculate how strong a password is.

As part of the Password Management system, when a new password is added or an existing one is edited, the user has the choice of either entering a password manually or let the application generate one. In order to generate a random password, a class was created to provide the feature and was added to `ECCryptoLib`.

The user can alter the password generator settings in `ECCrypto` so that the passwords generated can be according to the standards of the user. The user has the possibility of personalizing the following settings:

- The length of the generated password, by specifying the number of characters that the generated password should have;
- If it should contain lower case letters;
- If it should contain upper case letters;
- If it should contain numbers;
- If it should contain special characters and how many.

By default, the generated passwords contain upper and lower case letters, numbers, special characters, and the password length is of ten characters.

In order to ascertain how strong a password is and inform the user in an intuitive way, the following metric was used, which can be observed in table 5.1. The application uses a color scheme to explain how strong a password is by taking the following factors in consideration:

1. Password length is ten characters long or longer;
2. Password contains lower case letters;
3. Password contains upper case letters;
4. Password contains numbers;
5. Password contains special characters.

Color	Name	Description
Red	Very Weak	Password length is less than ten characters or there is no combination of types of characters (e.g., only letters or only numbers)
Orange	Weak	There is the combination of two different factors (e.g., upper and lower case letters are included)
Yellow	Good	There is the combination of three different factors (e.g., upper and lower case letters and numbers)
LimeGreen	Strong	There is the combination of four different factors (e.g., upper and lower case letters, numbers, and special characters)
Green	Very Strong	There is the combination of all the different factors (e.g., password includes upper and lower case letters, numbers, special characters and its length is at least ten characters long)

Table 5.1: Password generator strength metric table.

5.2.4 ECC methods

The implemented curves can be found in the `Curves` class. The Curves class holds the domain parameters for the NIST curves and can easily be extended to support other curves in the future.

The curves in table 5.2 are implemented in the library, although not all of them were used throughout the application. In order to try to ensure the best security possible using these algorithms, it was decided to only use the `P-521r1` curve in the application. It was also decided to use only randomly verifiable curves. The EC domain parameters that were used in the implementation can be found in Appendix A.

Parameters	Strength	Size	RSA/DSA
P-192r1	96	192	1536
P-224r1	112	224	2048
P-256r1	128	256	3072
P-384r1	192	384	7680
P-521r1	256	521	15360

Table 5.2: Representation of the implemented curves part of the `ECCryptoLib`.

While the `Curves` class holds the domain parameters for the curves, it is the `ECurve` class that handles all the logic of the curves, including the constructors, as well as the functions required to decode the points that are part of the domain parameters.

The `ECPoint` class contains all the properties, operations and methods that define a point in an elliptic curve. An elliptic curve point is defined by two `BigIntegers` `X` and `Y` which define the coordinates of the point in the curve. It also includes an instance of the curve for which the point is defined.

Using the formulae presented in 2.4.4.3 for the group operations on elliptic curves, methods to add two `ECPoints`, double an `ECPoint`, and multiply an `ECPoint` by a scalar were implemented and added to the class.

`ECAsymmetricKeyPair` is a class that represents an asymmetric key pair. An Asymmetric key pair in ECC is defined by a Ecurve, on which the private and public key are generated, a private key (a scalar) which is of the type `BigInteger`, and a public key which is a point in the curve.

The key pair is generated as follows:

1. Choose an elliptic curve E with modulus p;

   (a) Choose a random number d_A in $1, ..., n-1$ and calculate $X = (x_1.y_1) = d_A * G$;

   (b) Return $sk = d_a$ and $pk = (X, (p, a, b, G, n, h))$.

2. The encryption algorithm $E(pk, , m)$ works in the following way:

   (a) Choose a random number $d_B \in 1, ..., n-1$ and calculate $Y = (x_2, y_2) = d_B * G$;

   (b) Calculate $K = (x_3, y_3) = d_B * X$. The symmetric secret S between both entities is now x_3;

   (c) Use the KDF (see chapter 2) to derive two keys k_1 and k_2, one to encrypt, and the other to calculate the MAC;

   (d) Encrypt the message with k_1, i.e., $c \leftarrow E(k_1, m)$;

   (e) Calculate the MAC of the cryptogram with k_2, i.e., $t \leftarrow MAC(k_2, c)$;

   (f) Concatenate and return Y, the cryptogram and the MAC, i.e., (Y, c, t).

3. The decryption algorithm $D(sk, (Y, c, t))$ is as follows:

   (a) Derive the common point $K = (x_3, y_3) = d_B * Y$. The secret S between both parties is then x_3;

   (b) Use a KDF (see chapter 2) function to derive two keys k_1 and k_2, one to decrypt, and the other one to verify the MAC;

   (c) Verify MAC and fail if $t' \neq MAC(k_2, c)$;

   (d) Otherwise decrypt m from c using k_1, i.e., $m = D(k_1, c)$;

   (e) Return m.

The overall process described before results in the following:

1. The private key is a randomly generated number with the size of the curve field size.

2. The public key is calculated by multiplying the private key by the ECPoint G of the curve being used.

All the required logic to perform an ECDH key exchange is included in the `ECDH` class of the `ECCryptoLib`, which contain the source code for all the associated methods 2.4.5.

The `ECDSA` class contains the methods required to sign and verify signatures using the ECDSA. The ECIES class contains the methods required to encrypt and decrypt both messages and files using the `ECIES` and they were implemented as it was described in 2.4.6.

The library also includes extension methods that were necessary, such as a BigInteger extension method that calculates the ModInverse of two numbers using the extended euclidean algorithm, and which is not part of the .NET Framework BigInteger class. Included were Base64 encoding and decoding methods, among others.

5.3 ECCrypto Implementation

The ECCrypto project contains the WIndows Phone 8.1 application. The details of the project implementation will now be presented, organized by the different features that are provided with the application.

5.3.1 Using Sensors for Creating Randomness

Nowadays, smartphones typically include several sensors that are available to developers and allows them to create applications that take full advantage of their existence. These include applications that use the Global Positioning System (GPS) system of the phone for tracking the user location when performing different activities such as driving, running, walking and can even be used to show places of interest around the user. Gyroscope and accelerometers are very commonly used in games. When available, the application will try to use the following sensors for the purpose of generating passwords:

Gyrometer Sensor is a sensor that returns the angular velocity values with respect to the x, y, and z axes. Not all the phones have a gyrometer, so it is necessary to verify its existence. If the phone does have the sensor, then the values obtained can be used to increase the entropy of the password. In the `ECCrypto` app, if the gyrometer sensor is available, the values can be fetched when necessary.

Compass Sensor or magnetometer, can be used to determine the angle by which the device is rotated relative to the magnetic north pole. An application can also use raw magnetometer readings to detect magnetic forces around the device. If the sensor is available in the smartphone running `ECCrypto`, then its data will be used in the randomizing process.

Accelerometer Sensor measures the forces applied to the device at a moment in time. The direction that the user is moving the phone can be determined using these forces. The acceleration value is expressed as a 3-dimensional vector representing the acceleration components in the X, Y, and Z axes in gravitational units. The orientation of the acceleration is relative to the device such that -1g is applied in the Z-axis when the device is face up on a leveled table and -1g is applied to the Y-axis when the device is placed perpendicular to the table top. The Accelerometer sensor detects the force of gravity along with any forces resulting from the movement of the phone. The combined motion API, accessed using the `Motion` class, uses multiple device sensors to separate the gravity vector from the device acceleration and allows one to easily determine the current attitude (yaw, pitch, and roll) of the device. Given that not all smartphones are required to have an accelerometer sensor, it is required to check for its presence on the phone. If the sensor does exist, then the acceleration components in the X, Y and Z axe are used to increase the entropy in the system.

Inclinometer Sensor is a sensor that returns pitch, roll, and yaw values that correspond to rotation angles around x, y and z axes, respectively. If the sensor is available in the smartphone using the `ECCrypto` application, the values will be also be used.

Light Sensor Smartphones often have ambient-light sensors that are typically used to adapt the screen brightness according to the ambient light, in order to improve the screen viewing experience. This sensor is often available to developers and the values are returned as LUX values. `ECCrypto` will also use values of this sensor, if available.

Orientation Sensor is a sensor that it is mostly used to adjust the perspective of a user in a game. It returns a rotation matrix and a Quaternion. Quaternions add a fourth element to the [x, y, z] values that define a vector, resulting in arbitrary 4-D vectors. If the orientation sensor is available in the smartphone, then its values can and will be used in the `ECCrypto` application.

5.3.2 Unlock ECCrypto App

On the first time that `ECCrypto` is started, the user is prompted to choose a numeric password to protect the access to the application. It is an optional setting, but it is advisable that a password is set in order to guarantee that all the sensitive information contained in the app remains secure. Once the password has been set, every time the application is started, the user needs to input the password using the keyboard that is provided.

The unlock keyboard is not the standard keyboard, therefore it is not related to the system keyboard nor is it affected by its events. It was created in order to ensure a more secure experience. The keys that are pressed on the keyboard are not highlighted in opposite to most keyboards. This decision may affect the feedback to the user regarding the key that was pressed, but it is a safer approach because this way, someone from afar will have a harder time trying to figure out the key that was pressed. The keyboard is also randomly generated every time a key is pressed meaning that, once again, if someone is close enough to see the positions of the keys but not their numbers, the person will not be able to infer which numbers were being pressed. The application does not give any kind of feedback to the user regarding the password being incorrect. This means that, if the inputed password is correct, then the application will go straight to the main page, but if it is not, the user will not see any kind of message. Applications in Windows Phone often display the last entered character of the password as it is being typed but then hide it after a short amount of time. This feature was not added so that other people cannot get a glimpse of the password.

It is to be noted that if a password is set, the password cannot be recovered, so the user will effectively lose access to the application since it is necessary to enter the application. The password can be enabled, disabled and altered in the settings page of `ECCrypto`.

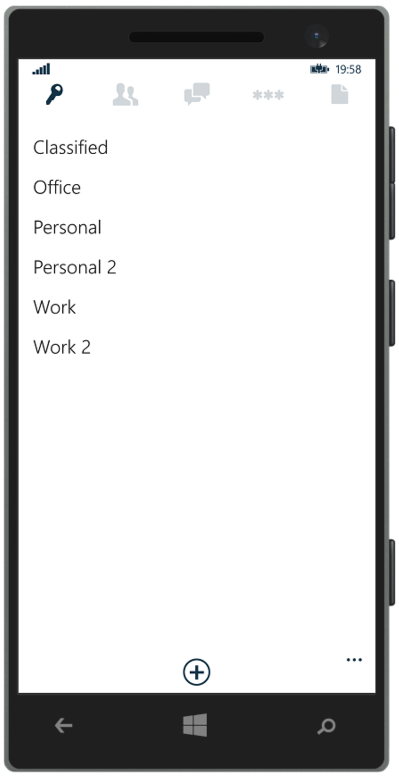
5.3.3 Trust Management System

The trust management system is used to ensure that contacts are who they say to be. A contact key may not be added via QR code exchange, but by simply pasting the key on the appropriate public key field of the application. When this happens, there is no guarantee that the exchanged key truly belongs to the contact because the channel through which the key was exchanged may not be secure. In order to verify a key that was not exchanged via QR code, the user must ask other contacts that also have the key to verify if it is a trusted key. The number of people that must confirm a key before it is considered fully trusted can be changed in the settings of the application. While the trust management system does not directly affect the overall use of the application, it does give more security to the users in knowing that the exchanged keys are secure. In an optimal scenario, a user would only be able to exchange messages with contacts whose keys are fully trusted. However, that limitation would likely lead most users not to use the application.

5.3.4 Application Overview

The Home page is the main page of the application and it is where the user navigates to when the application is unlocked. It is from the Home page that the user can access all the features of the application.

Figure 5.1 shows the list of user keys, in the home page of the application, that were generated and added to `ECCrypto`. As for figure 5.2, it is the screen in which the user can add a new user


Classified

Office

Personal

Personal 2

Work

Work 2

Figure 5.1: Screenshot of the home page showing the list of the user keys.

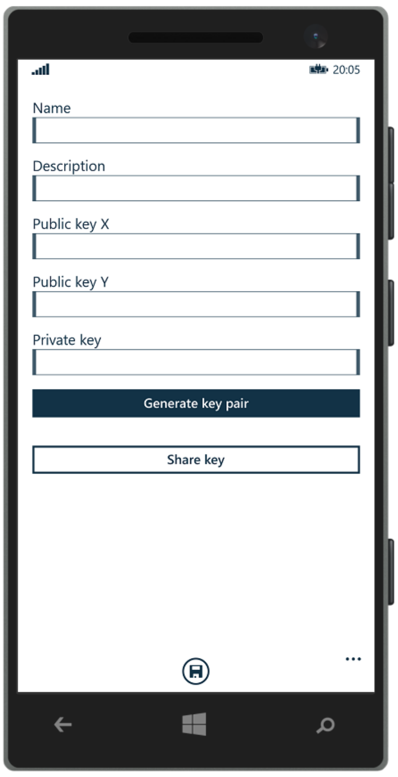


Figure 5.2: Screenshot of the screen where the user can add new keys.

key. This is also the screen where the new keys are generated.

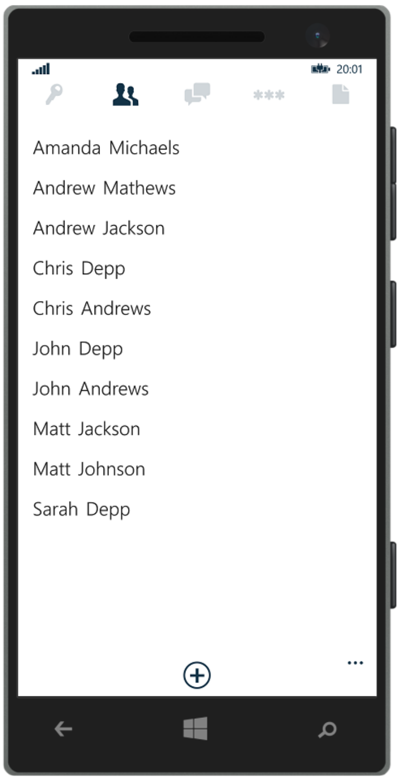
Figure 5.3 shows the home screen which displays a list of the contacts that the user has added. In figure 5.4 one may see detailed view in which a user can add or edit a contact, including adding phone numbers, emails or public keys for the contact.

All the messages that were encrypted/decrypted by the application can be seen in figure 5.5. Figure 5.6 shows the menu that can be seen from the bottom, which is shown in all the tabs of the home page and gives access to the application settings and the about page.

Figure 5.7 shows the message detail page when a user selects a message from the home page. From this page, the user can choose to reply to the message, delete the message or go back to the home page. The page used to send a new message can be seen in figure 5.8.

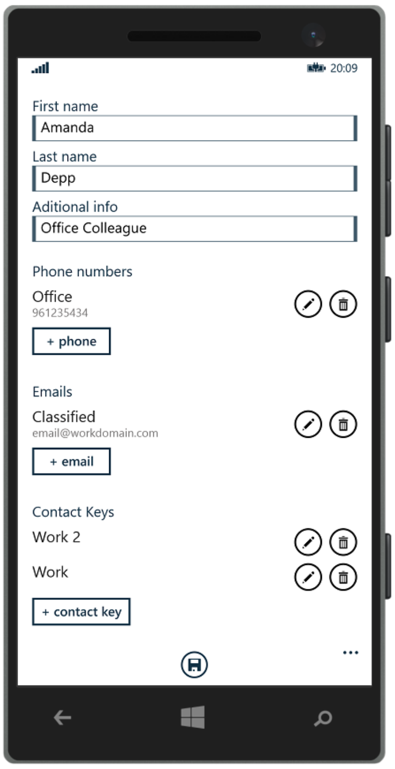
In figure 5.9 it is possible to see the home page where a user can see a list of all the added passwords and can add a new one. Figure 5.10 shows the page where a user can edit or delete a password. In both pages, the user has access to the password generator settings.

Figure 5.11 shows the home page tab where the user can choose to encrypt or decrypt a file. As for figure 5.12, it displays the settings screen, where the user can change different settings in the `ECCrypto` application, such as setting the password to lock the application, or in the


20:01

Amanda Michaels
Andrew Mathews
Andrew Jackson
Chris Depp
Chris Andrews
John Depp
John Andrews
Matt Jackson
Matt Johnson
Sarah Depp

Figure 5.3: Screenshot of the home page showing the list of the contacts of the user.


20:09

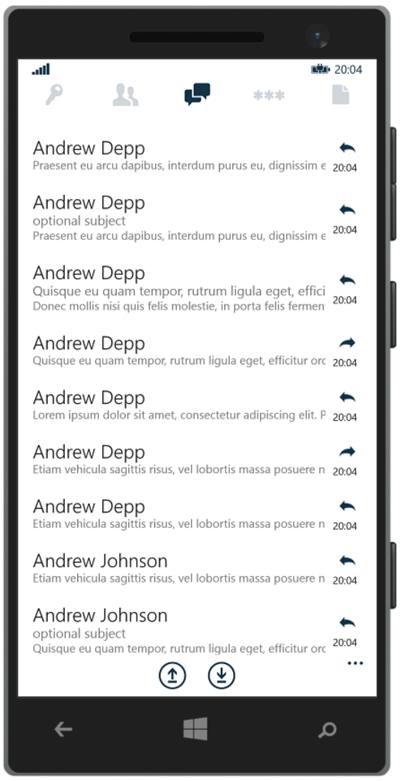
First name: Amanda
Last name: Depp
Aditional info: Office Colleague

Phone numbers
Office
961235434

Emails
Classified
email@workdomain.com

Contact Keys
Work 2
Work

Figure 5.4: Screenshot of the page where a contact can be edited.


20:04

Andrew Depp
Praesent eu arcu dapibus, interdum purus eu, dignissim e
20:04

Andrew Depp
optional subject
Praesent eu arcu dapibus, interdum purus eu, dignissim e
20:04

Andrew Depp
Quisque eu quam tempor, rutrum ligula eget, effici
Donec mollis nisi quis felis molestie, in porta felis fermen
20:04

Andrew Depp
Quisque eu quam tempor, rutrum ligula eget, efficitur orc
20:04

Andrew Depp
Lorem ipsum dolor sit amet, consectetur adipiscing elit. P
20:04

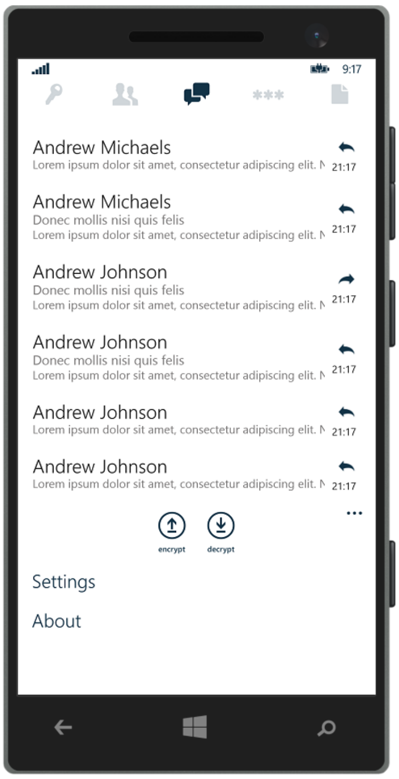
Andrew Depp
Etiam vehicula sagittis risus, vel lobortis massa posuere n
20:04

Andrew Depp
Etiam vehicula sagittis risus, vel lobortis massa posuere n
20:04

Andrew Johnson
Etiam vehicula sagittis risus, vel lobortis massa posuere n
20:04

Andrew Johnson
optional subject
Quisque eu quam tempor, rutrum ligula eget, efficitur orc
20:04

Figure 5.5: Screenshot of the home page showing the list of the messages that have been encrypted and decrypted by the user.


9:17

Andrew Michaels
Lorem ipsum dolor sit amet, consectetur adipiscing elit. N
21:17

Andrew Michaels
Donec mollis nisi quis felis
Lorem ipsum dolor sit amet, consectetur adipiscing elit. N
21:17

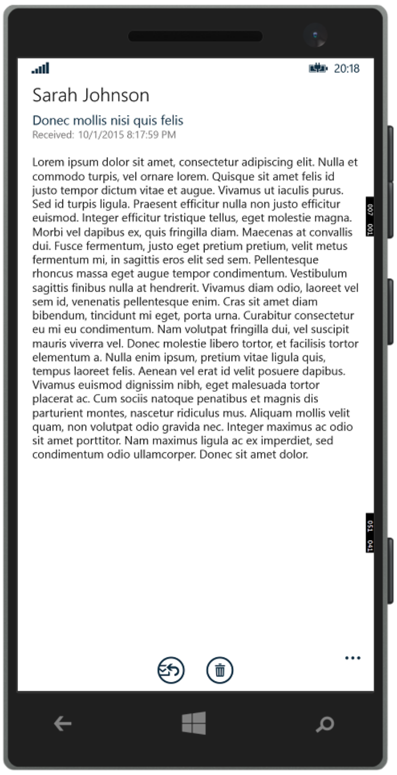
Andrew Johnson
Donec mollis nisi quis felis
Lorem ipsum dolor sit amet, consectetur adipiscing elit. N
21:17

Andrew Johnson
Donec mollis nisi quis felis
Lorem ipsum dolor sit amet, consectetur adipiscing elit. N
21:17

Andrew Johnson
Lorem ipsum dolor sit amet, consectetur adipiscing elit. N
21:17

Andrew Johnson
Lorem ipsum dolor sit amet, consectetur adipiscing elit. N
21:17

Figure 5.6: Screenshot showing the application bar that gives access to the settings of the application.


20:18

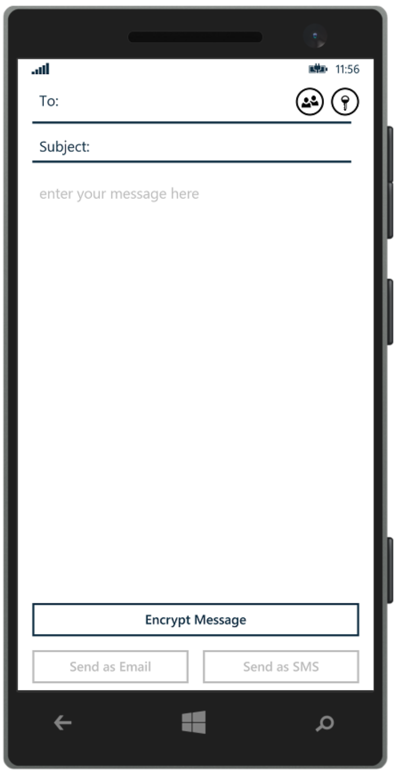
Sarah Johnson

Donec mollis nisi quis felis

Received: 10/1/2015 8:17:59 PM

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et commodo turpis, vel ornare lorem. Quisque sit amet felis id justo tempor dictum vitae et augue. Vivamus ut iaculis purus. Sed id turpis ligula. Praesent efficitur nulla non justo efficitur euismod. Integer efficitur tristique tellus, eget molestie magna. Morbi vel dapibus ex, quis fringilla diam. Maecenas at convallis dui. Fusce fermentum, justo eget pretium pretium, velit metus fermentum mi, in sagittis eros elit sed sem. Pellentesque rhoncus massa eget augue tempor condimentum. Vestibulum sagittis finibus nulla at hendrerit. Vivamus diam odio, laoreet vel sem id, venenatis pellentesque enim. Cras sit amet diam bibendum, tincidunt mi eget, porta urna. Curabitur consectetur eu mi eu condimentum. Nam volutpat fringilla dui, vel suscipit mauris viverra vel. Donec molestie libero tortor, et facilisis tortor elementum a. Nulla enim ipsum, pretium vitae ligula quis, tempus laoreet felis. Aenean vel erat id velit posuere dapibus. Vivamus euismod dignissim nibh, eget malesuada tortor placerat ac. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam mollis velit quam, non volutpat odio gravida nec. Integer maximus ac odio sit amet porttitor. Nam maximus ligula ac ex imperdiet, sed condimentum odio ullamcorper. Donec sit amet dolor.

Figure 5.7: Screenshot of the message page after the user selects a message from the list.


11:56

To:

Subject:

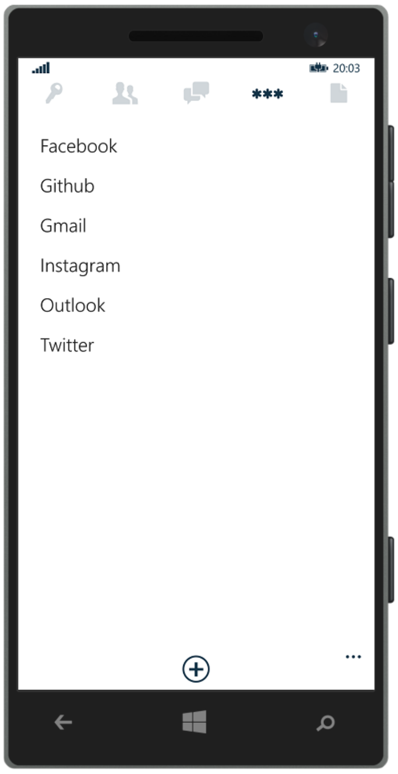
enter your message here

Encrypt Message

Send as Email

Send as SMS

Figure 5.8: Screenshot of the page in which the user can send a new message.


20:03

***

Facebook

Github

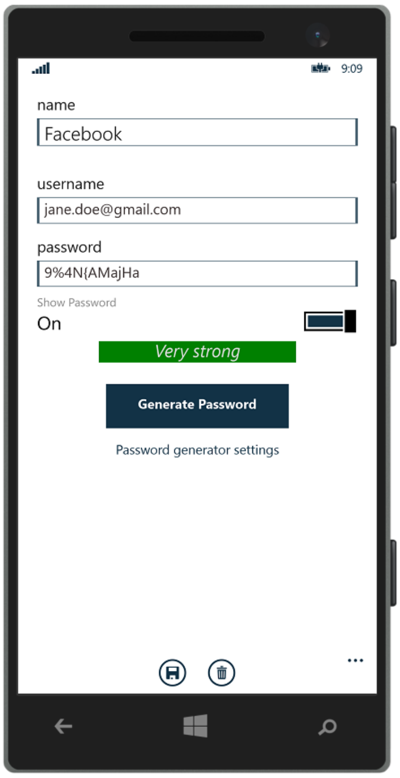
Gmail

Instagram

Outlook

Twitter

Figure 5.9: Screenshot of the home page showing the list of the passwords of the user.


9:09

name

Facebook

username

jane.doe@gmail.com

password

9%4N{AMajHa

Show Password

On

Very strong

Generate Password

Password generator settings

Figure 5.10: Screenshot of the detailed view of a password.

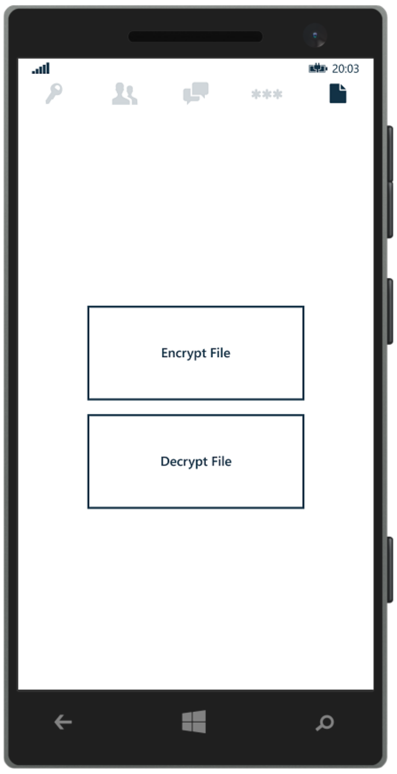


Figure 5.11: Screenshot of the home page where the user can encrypt/decrypt files.

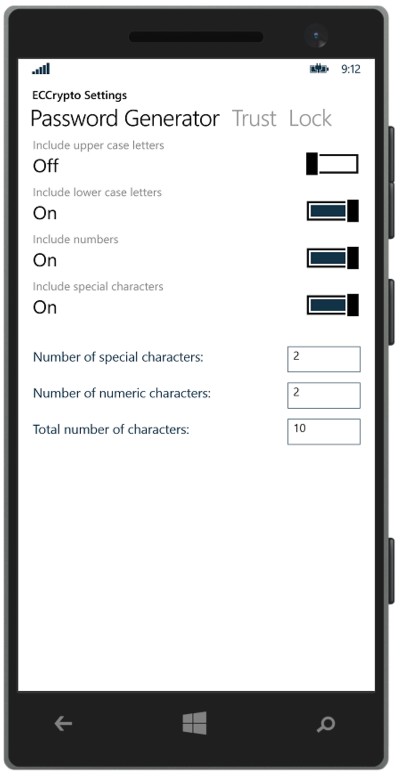

9:12
ECCrypto Settings
Password Generator Trust Lock
Include upper case letters
Off
Include lower case letters
On
Include numbers
On
Include special characters
On
Number of special characters: 2
Number of numeric characters: 2
Total number of characters: 10

Figure 5.12: Screenshot of `ECCrypto` settings for the password generator.

currently displayed tab, the password generator settings.

Figures 5.13 and 5.14 show the interface where the user can input a password and unlock the application to then move to the main page of the application. The random keyboard functionality, was previously described in section 5.3.2. The user can add a key for the contact by either copying and pasting the key (which will automatically be considered an untrusted key) or import the key using the smartphone camera and QR codes. Figures 5.15, 5.16 show the different screens used during the process of the key exchange via QR code.

The process of a key exchange requires both users to be present in the same place and use their phones to exchange public keys. One user needs to select the option in the application to share a user key. The other user needs to add the new key to the keys of the contact by selecting the import key option. The required steps to share a key with a contact are as follows:

1. User A selects the option `Share Key` and a QR code is generated and displayed on the screen;
2. User B selects the option `Import Key` and the camera on the phone is started. User B scans the QR code:
3. User B generates a random string, a QR code is generated and shown to User A;

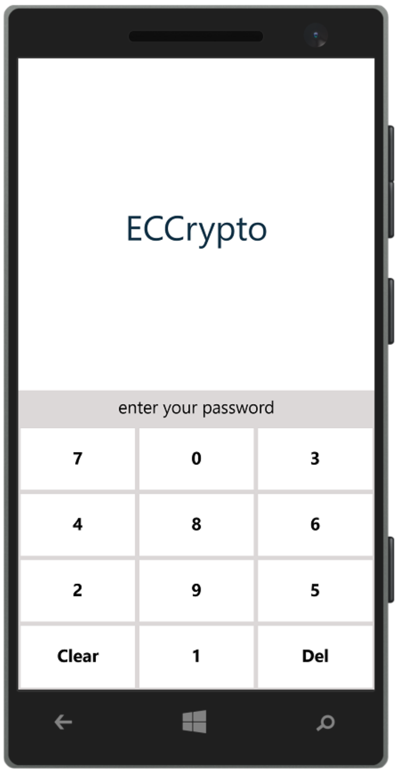


Figure 5.13: Screenshot of the Unlock page with randomized keyboard.

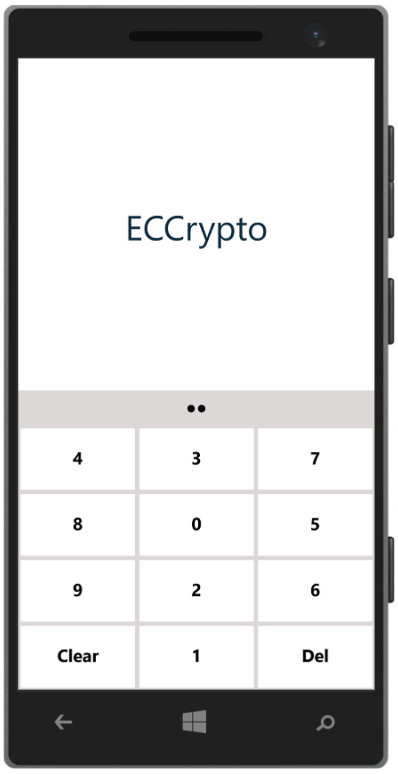


Figure 5.14: Screenshot of the Unlock page with alternate randomized keyboard.

4. User A scans the QR code, signs it with the exchanged key and shows a new QR code with the signature;

5. User B scans the QR code containing the signature and verifies it. If the signature is valid, then the key is added. Otherwise it informs the user that an error has occurred.

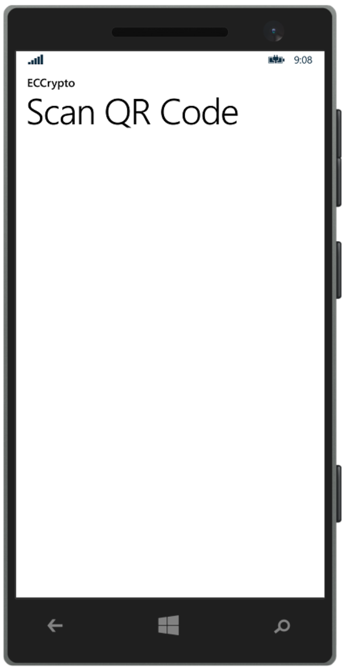


Figure 5.15: Screenshot of the page in which a QR code is scanned.

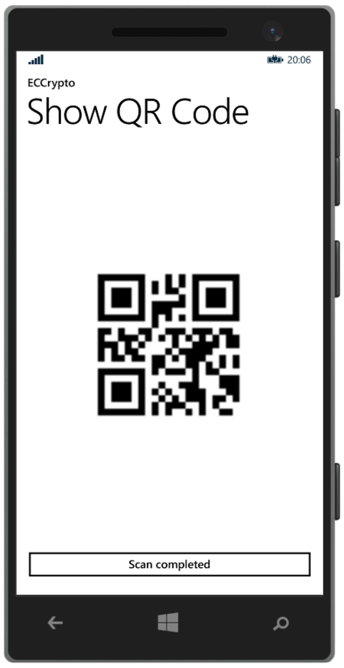


Figure 5.16: Screenshot of the page in which a QR code is displayed.

5.4 Testing of the Prototype

In this section, the testing of the `ECCrypto` application and of the `ECCryptoLib` will be discussed.

5.4.1 ECCrypto

During the development of `ECCrypto`, it was necessary to test the application as the different features were added. Visual studio includes various emulators that can run the application so that it can be tested. However, testing only in an emulator is rarely enough to fully test an application. Thus, the application was initially tested using the available emulators and afterwards was tested in physical devices.

Emulator

The Windows Phone Emulator is a desktop application that emulates a Windows Phone device. It provides a virtualized environment in which a developer can test and debug Windows Phone apps without having a physical device.

During the initial phase of the implementation, when most of the development was design related, the Visual Studio emulators were used to test and debug the application. There are several emulators available that allow to test the application for different resolution and screen factors. Throughout the development of the application the following emulators were used:

- Windows Phone Emulator 8.1 WVGA 4 inch;

- Windows Phone Emulator 8.1 WVGA 4 inch 512MB;
- Windows Phone Emulator 8.1 WXGA 4.5 inch;
- Windows Phone Emulator 8.1 720p 4.7 inch;
- Windows Phone Emulator 8.1 1080p 5.5 inch;
- Windows Phone Emulator 8.1 1080p 6 inch.

Windows Phone

Even though much of the development can be done using the Windows Phone Emulators, it is always recommended to test the application in physical devices in order to ensure that every aspect of the application works well in a real environment where there are other factors that can impact the application. Even though emulators include the possibility to test some sensors, it is never as reliable or simple as testing the application in a physical device. Not only that, but the key exchange feature needs to be tested in two physical devices, because it requires the use of the camera of the phone in order to scan a QR code that is shown on the screen of the other phone.

`ECCrypto` was tested using the three different smartphones: Nokia Lumia 720, Nokia Lumia 930 and Samsung Ativ S i8750. The main specifications of the phones can be found in table 5.3

Phone Model	Processor	Memory	Screen size	Year
Nokia Lumia 720	Dual-core 1 GHz	512	4.3” (480 x 800 pixels)	2013
Samsung Ativ S I8750	Dual-core 1.5 GHz Krait	1024	4.8” (720 x 1280 pixels)	2012
Nokia Lumia 930	Quad-core 2.2 GHz Krait 400	2048	5” (1080 x 1920 pixels)	2014

Table 5.3: Specifications for the phones used during the testing of the application.

5.4.2 ECCryptoLib

During the implementation of the `ECCryptoLib`, it was often necessary to test the implemented cryptographic methods, regardless of the fact that the Windows Phone application was not yet finished. This requirement led to the creation of a unit testing project that allows testing the library without having to have implemented the logic in the application that would use the methods being tested.

A project named `ECCryptoLibTest` was added to the solution and tests for the implemented cryptographic methods were added. The password generation and verification methods were tested to ensure that the different generator settings produced the correct results. ECC key generation tests were also added in which, for instance, 100 keys were generated and verified if they were correctly generated. ECDH and ECDSA methods were also tested, in order to guarantee that the protocols were working correctly. Table 5.4 lists all the unit tests performed to the library and table 5.5 describes the tests.

Test ID	Test Name	Result
1	AES file encryption and decription	Passed
2	Base64 encoding and decoding	Passed
3	SHA-1 calculation	Passed
4	SHA-256 calculation	Passed
5	SHA-384 calculation	Passed
6	SHA-512 calculation	Passed
7	MD5 calculation	Passed
8	HMAC-SHA-1 calculation	Passed
9	HMAC-SHA-256 calculation	Passed
10	HMAC-SHA-384 calculation	Passed
11	HMAC-SHA-512 calculation	Passed
12	HMAC-MD5 calculation	Passed
13	Multiplication of EC Point by a scalar operation test	Passed
14	EC key pair generation for NIST P192r1 curve	Passed
15	EC key pair generation for NIST P224r1 curve	Passed
16	EC key pair generation for NIST P256r1 curve	Passed
17	EC key pair generation for NIST P384r1 curve	Passed
18	EC key pair generation for NIST P521r1 curve	Passed
19	ECDH using keys generated by NIST P192r1 curve	Passed
20	ECDH using keys generated by NIST P224r1 curve	Passed
21	ECDH using keys generated by NIST P256r1 curve	Passed
22	ECDH using keys generated by NIST P384r1 curve	Passed
23	ECDH using keys generated by NIST P521r1 curve	Passed
24	ECDSA using keys generated by NIST P192r1 curve	Passed
25	ECDSA using keys generated by NIST P224r1 curve	Passed
26	ECDSA using keys generated by NIST P256r1 curve	Passed
27	ECDSA using keys generated by NIST P384r1 curve	Passed
28	ECDSA using keys generated by NIST P521r1 curve	Passed
29	ECIES using keys generated by NIST P521r1 curve	Passed

Table 5.4: Results of the unit tests performed on the `ECCryptoLib`.

Test ID	Test Description
1	Decrypted plaintext matches the previously encrypted plaintext
2	Decoded string matches string before being encoded
3	SHA-1 calculation matches expected value for the string
4	SHA-256 calculation matches expected value for the string
5	SHA-384 calculation matches expected value for the string
6	SHA-512 calculation matches expected value for the string
7	MD5 calculation matches expected value for the string
8	HMAC-SHA-1 calculation matches expected value for the message
9	HMAC-SHA-256 calculation matches expected value for the message
10	HMAC-SHA-384 calculation matches expected value for the message
11	HMAC-SHA-512 calculation matches expected value for the message
12	HMAC-MD5 calculation matches expected value for the message
13	Result of the multiplication of EC Point by a scalar `n` is equal to the sum of the point with itself `n` times
14	Generated key is valid if the public key is equal to the multiplication of the private key for the `G` point
15	Generated key is valid if the public key is equal to the multiplication of the private key for the `G` point
16	Generated key is valid if the public key is equal to the multiplication of the private key for the `G` point
17	Generated key is valid if the public key is equal to the multiplication of the private key for the `G` point
18	Generated key is valid if the public key is equal to the multiplication of the private key for the `G` point
19	Result of the ECDH exchange match
20	Result of the ECDH exchange match
21	Result of the ECDH exchange match
22	Result of the ECDH exchange match
23	Result of the ECDH exchange match
24	Signature using ECDSA is generated for a string and it is successfully verified
25	Signature using ECDSA is generated for a string and it is successfully verified
26	Signature using ECDSA is generated for a string and it is successfully verified
27	Signature using ECDSA is generated for a string and it is successfully verified
28	Signature using ECDSA is generated for a string and it is successfully verified
29	Decrypted plaintext matches plaintext prior to being encrypted

Table 5.5: Description of the unit tests performed on the `ECCryptoLib`.

5.5 ECCrypto Submission

The `ECCrypto` application was submitted to the Windows Store through the dashboard available to developers. The submission process included adding a description, screenshots, keywords, as well as choosing the rating of the application in different countries. The application was made available in all markets in the world, free of charge to the users. The application can be downloaded from the Windows Store. As for the library, `ECCryptoLib` is available on GitHub [Ana15], as a public repository. Fork requests are allowed so that it can be improved by other users that wish to contribute. The library was released under an MIT license.

5.6 Conclusion

This chapter started by discussing the implementation of the `ECCryptoLib`. The `ECCryptoLib` contains all the cryptographic methods used by the application. These methods include hashing algorithms, password generation, AES encryption and decryption. All of the ECC methods are also part of the library. The library was made available on GitHub, where other users may contribute if they wish to.

The `ECCrypto` application was analyzed and the implementation of the main features of the application were explained. As a result of this work, the `ECCrypto` application was submitted to the Windows Store. The discussion and presentation of features included in this chapter demonstrate that the objectives set for this master's program and for the application, during the engineering phase, were successfully achieved.

Chapter 6

Conclusions and Future Work

In this chapter, the main conclusions of this work will be presented. The future work possibilities regarding `ECCrypto` will also be discussed.

6.1 Conclusions

It is undeniable that these are times of great insecurity in terms of information security and privacy. Several incidents that have occurred in the last few years have shone a spotlight in this issue by people like Edward Snowden, Chelsea Manning and Julian Assange which came out in public with confidential information.

`ECCrypto` is an application for Windows Phone that helps ensure that a user can communicate in a secure way. It contains a set of functionalities that are commonly found in separate applications, and brings them all together into a single application. It also uses stronger cryptographic functions than the ones that are typically found in the available applications. `ECCrypto` features include the storage and management of passwords of the user and it includes a random password generator. It supports the encryption and decryption of messages and it easily incorporates the functionality of sending an encrypted message via SMS or email. It also allows encrypting and decrypting files stored in the smartphone. User keys can be generated, stored and used by the user in communications. These are generated using ECC and true randomness sources such as the sensors present in the smartphone. The user can manage contacts and their public keys. They key exchange process uses QR codes to facilitate the exchanging process. A trust management system was created to help ensure the authenticity of the keys of the contacts, in which the keys can be verified by other contacts, in order to guarantee that the key belongs to the user.

All these functionalities use ECC, via the generation of user keys using elliptic curves and by using protocols like ECDH to perform secure key exchanges and ECDSA for digitally signing messages and files. Even though ECC is not very commonly used nowadays, it is a viable replacement for RSA, which may be broken in the next few years and has advantages such as smaller key sizes, while providing stronger security.

The ECC functions were all implemented from scratch because, at the time of writing this dissertation, there were no available libraries including the necessary functions. The library, ECCryptoLib, includes the implementation of the elliptic curves and necessary operations, generation of public-private key pairs, methods for ECDH and ECDSA. Functions for encrypting and decrypting messages and files using AES were also included. A random password generator was implemented and several hashing algorithms were also made available to anyone that imports the library.

Nowadays if a user truly wishes to maintain information private and confidential he has to resort to applications that provide this type of functionalities because they are not available by default in the current systems. Information privacy and security should be enabled by design but it is not and it can have a serious impacts in the life of people when information leaks occur. ECCrypto was created to help users, in different fields, to communicate in a secure way.

The objectives defined for this master's program were successfully achieved. The main objective described in section 1.2 was achieved by the implementation of the application; the security was assured by designed due to the software engineering process, testing and implementation of the library (though further tests are needed).

6.2 Future Work

The main functionalities of ECCrypto have been implemented but the application can be improved in several aspects. Even though the ECCryptoLib has several elliptic curves implementations, only the 512bit curve is being used in the ECCrypto. This decision was due to the complexity that would be added to the key exchange process and to the key management from the perspective of the user, as it would become more complex and it needs to be done transparently to the user. As such, it should be thoroughly analyzed in the future.

The application should be tested for usability, mostly because of the number of features provided due to its complexity and dimension, which may be somewhat confusing for a typical user that does not grasp the concept of the application. It is also necessary to ensure user retention so that, when users install the application, they find it easy to work with and actually use it. This could be achieved for instance by using tools like Mixpanel [Mix15] to find patterns in common user actions and also areas that can be improved.

It would be interesting to include other standard elliptic curves such as the Brainpool curves, in the ECCryptoLib, so that users are not limited to using the NIST curves. The library can easily support the implementation of new curves.

It is important to implement application versions for other mainstream mobile OS, such as Android and iOS so that people using different platforms can all communicate with each other. An upgrade for Windows 10 is also advisable, since it will allow the application to run in both Windows Phones, desktops, tablets and other devices of the Windows family.

Another feature that would be important to include is the ability to sync information on different devices. For instance, if a user has a smartphone with `ECCrypto` all set up with keys and contacts, it seems important that the user should be able to get the same information on a tablet or laptop.

Bibliography

[Ana15] Ana Figueira. ECCryptoLib [online]. 2015. Available from: `https://github.com/AnaFigueira/ECCryptoLib` [cited October 4 , 2015]. xvii, 73

[Anc15] Ancylo. Password Jinni [online]. 2015. Available from: `https://www.microsoft.com/en-us/store/apps/password-jinni/9nblggh0b2ll` [cited September 17, 2015]. 11

[App12] Apps4Life. Secure Texting [online]. 2012. Available from: `https://itunes.apple.com/tr/app/secure-texting-password-protect/id528988546?mt=8` [cited September 17, 2015]. 10

[App15] Apple Inc. iTunes [online]. 2015. Available from: `https://www.apple.com/itunes` [cited September 17, 2015]. xi, 9

[Are15] Arenim Technologies. CryptTalk Pro [online]. 2015. Available from: `https://itunes.apple.com/pt/app/crypttalk-pro/id892554609?mt=` [cited September 17, 2015]. 10

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In PhongQ. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 1-16. Springer Berlin Heidelberg, 2014. Available from: `http://dx.doi.org/10.1007/978-3-642-55220-5_1`. 23

[Cho] Wendy Chou. Elliptic curve cryptography and its applications to mobile devices. Available from: `http://honors.cs.umd.edu/reports/ECCpaper.pdf`. 2

[Cis] Cisco. Cisco 2014 Annual Security Report. Available from: `https://www.cisco.com/web/offer/gist_ty2_asset/Cisco_2014_ASR.pdf`. 54

[Cov15] CoverMe, Inc. CoverMe Private Texting [online]. 2015. Available from: `https://itunes.apple.com/us/app/coverme-private-texting-messenger/id593652484?mt=8` [cited September 17, 2015]. 10

[Cri15] Criptyque. Pryvate - The Encryption App [online]. 2015. Available from: `https://itunes.apple.com/us/app/pryvate-the-encryption-app/id914948581?mt=8` [cited September 17, 2015]. 9

[Cyb15] CyberWorm. Crypter [online]. 2015. Available from: https://www.microsoft.com/en-us/store/apps/crypter/9nblggh09wvl [cited September 17, 2015]. 14

[Goo15] Google. Android Play Store [online]. 2015. Available from: https://play.google.com/store/apps [cited September 17, 2015]. xi, 8

[GPW+04] Nils Gura, Arun Patel, Arvinderpal W, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. pages 119–132, 2004. 2

[J.M15] J.M. Porup. This New 'Secure' App for Journalists May Not Be Secure At All [online]. 2015. Available from: http://motherboard.vice.com/read/this-new-secure-app-for-journalists-may-not-be-secure-at-all [cited October 2, 2015]. 1

[JOP14] Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. The past, evolving present, and future of the discrete logarithm. In Çetin Kaya Koç, editor, *Open Problems in Mathematics and Computational Science*, pages 5–36. Springer International Publishing, 2014. Available from: http://dx.doi.org/10.1007/978-3-319-10683-0_2. 23

[Kas] Kaspersky Lab and INTERPOL Joint Report. Mobile Cyber Threats. Available from: http://media.kaspersky.com/pdf/Kaspersky-Lab-KSN-Report-mobile-cyberthreats-web.pdf [cited September 21, 2015]. 8

[Kas12] Kaspersky. Android Mobile Security Threats [online]. 2012. Available from: https://usa.kaspersky.com/internet-security-center/threats/mobile#.Vavsj_lVgk0 [cited September 17 , 2015]. 8

[Kob] Neal Koblitz. Elliptic Curve Cryptosystems. In *Mathmatics of computation*, number 177, pages 203–209. Available from: http://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866109-5/S0025-5718-1987-0866109-5.pdf. 22, 29

[LFr15] LFramCompany. TXTCrypt [online]. 2015. Available from: https://www.microsoft.com/en-us/store/apps/txtcrypt/9nblggh0ft4j [cited September 17, 2015]. 12

[Lun15] LunaPlena. Secure Text Pro [online]. 2015. Available from: https://www.microsoft.com/en-us/store/apps/secure-text-pro/9nblggh0gr63 [cited September 17, 2015]. 11

[M. 10] M. Lochter and J. Merkle. Elliptic Curve Cryptography (ECC) Brainpool Standard

Curves and Curve Generation [online]. 2010. Available from: https://tools.ietf.org/html/rfc5639 [cited April 27, 2015]. 28

[Mic] Microsoft. Windows Phone 8.1 Security Overview. Available from: https://www.microsoft.com/en-us/download/details.aspx?id=42509. 53

[Mix15] Mixpanel. Mixpanel [online]. 2015. Available from: https://mixpanel.com/ [cited September 29, 2015]. 76

[Nat09] National Security Agency. Cryptography Today [online]. 2009. Available from: https://www.nsa.gov/ia/programs/suiteb_cryptography/ [cited April 12, 2015]. 28

[num93] The number field sieve. In ArjenK. Lenstra and Jr. Lenstra, HendrikW., editors, *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics*. 1993. Available from: http://dx.doi.org/10.1007/BFb0091537. 23

[Ope15] Open Source Initiative. The MIT License (MIT) [online]. 2015. Available from: https://opensource.org/licenses/MIT [cited October 1, 2015]. xi, 5

[oSN] National Institute of Standards and Technology (NIST). Announcing the ADVANCED ENCRYPTION STANDARD (AES) . Available from: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. 21

[Par15] Paranoia Works. Secret Space Encryptor (SSE) - Universal Encryption App [online]. 2015. Available from: https://play.google.com/store/apps/details?id=com.paranoiaworks.unicus.android.sse [cited September 17, 2015]. 9

[Pau] Paul Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*. 20

[Pau13] Paul Bakker. Elliptic Curve performance: NIST vs Brainpool [online]. 2013. Available from: https://tls.mbed.org/kb/cryptography/elliptic-curve-performance-nist-vs-brainpool [cited September 14 , 2015]. 30

[PLR15] PLRS. Text Encryption Pro [online]. 2015. Available from: https://www.microsoft.com/en-us/store/apps/text-encryption-pro/9nblggh0fgvw [cited September 17, 2015]. 12

[Ram15] Ramon Llamas, Ryan Reith and Kathy Nagamine. Smartphone OS Market Share, 2015 Q2 [online]. 2015. Available from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp [cited September 29, 2015]. 8

[Res] Certicom Research. Sec 1: Elliptic curve cryptography. Available from: http://cs.ucsb.edu/~koc/cren/docs/w03/sec1_final.pdf. 28, 29

[Ron15] Ronca. EnkrypTo [online]. 2015. Available from: https://www.microsoft.com/en-us/store/apps/enkrypto/9nblggh07t5q [cited September 17, 2015]. 13

[Ror] Rorot. ECC: A Case for Mobile Encryption [online]. Available from: http://resources.infosecinstitute.com/ecc-case-mobile-encryption/. 2

[See15] Seecrypt. Seecrypt SC3 [online]. 2015. Available from: https://itunes.apple.com/us/app/seecrypt-sc3/id889036096?mt=8 [cited September 17, 2015]. 10

[Sil13] Silent Circle. Silent Text [online]. 2013. Available from: http://gizmodo.com/silent-text-for-android-a-secure-text-service-that-sel-1253727782 [cited September 17, 2015]. 8

[SMS15] SMSSecure. SMSSecure [online]. 2015. Available from: https://play.google.com/store/apps/details?id=org.smssecure.smssecure [cited September 17, 2015]. 9

[Sta] Williams Stallings. *Cryptography and Network Security - Principles and Practice*. Pearson Education (US). 2

[Tom13] Tom Simonite. Math Advances Raise the Prospect of an Internet Security Crisis [online]. 2013. Available from: http://www.technologyreview.com/news/517781/math-advances-raise-the-prospect-of-an-internet-security-crisis/ [cited September 23, 2015]. ix, 2

[Vic] Victor S. Miller. Use of Elliptic Curves in Cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 417–426. 22

[Wha] Ryan Edward Whaley. Algorithms for solving the discrete logarithm problem. Available from: http://encompass.eku.edu/etd/235/. 23

[woo14] woodkick. ECC SMS [online]. 2014. Available from: https://play.google.com/store/apps/details?id=com.woodkick.eccsms [cited September 17, 2015]. 8

Appendix A

EC Domain Parameters

This appendix contains EC domain parameters for the curves that were implemented in the `ECCryptoLib`, as mentioned in subsection 5.2.4.

A.1 Recommended Parameters for secp192r1

The elliptic curve domain parameters over $\mathbb{F}_p$ associated with a Koblitz curve `secp192r1` are specified by the sextuple $T = (p, a, b, G, n, h)$ where the finite field $\mathbb{F}_p$ is defined by:

$$p = \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF} \\ = 2^{192} - 2^{64} - 1$$

The curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

$$a = \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFC}, \\ b = \texttt{64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1},$$

E was chosen verifiably at random from the seed:

$$S = \texttt{3045AE6F C8422F64 ED579528 D38120EA E12196D5},$$

The base point *G* in compressed form is:

$$G = \texttt{03 188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012},$$

Finally the order n of G and the cofactor are:

$$n = \texttt{FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831}, \\ h = \texttt{01}.$$

A.2 Recommended Parameters for secp224r1

The elliptic curve domain parameters over $\mathbb{F}_p$ associated with a Koblitz curve `secp224r1` are specified by the sextuple $T = (p, a, b, G, n, h)$ where the finite field $\mathbb{F}_p$ is defined by:

$$p = \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 00000000 00000001} \\ = 2^{224} - 2^{96} + 1$$

The curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

$a =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFE

$b =$ B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4

E was chosen verifiably at random from the seed:

$S =$ BD713447 99D5C7FC DC45B59F A3B9AB8F 6A948BC5

The base point *G* in compressed form is:

$G =$ 02 B70E0CBD 6BB4BF7F 321390B9 4A03C1D3 56C21122 343280D6 115C1D21

Finally the order n of G and the cofactor are:

$n =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFF16A2 E0B8F03E 13DD2945 5C5C2A3D

$h =$ 01

A.3 Recommended Parameters for secp256r1

The elliptic curve domain parameters over $\mathbb{F}_p$ associated with a Koblitz curve `secp256r1` are specified by the sextuple $T = (p, a, b, G, n, h)$ where the finite field $\mathbb{F}_p$ is defined by:

$p =$ FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF

$$= 2^{224}(2^{32} - 1) - 2^{192} + 2^{96} - 1$$

The curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

$a =$ FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC

$b =$ 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B

E was chosen verifiably at random from the seed:

$S =$ C49D3608 86E70493 6A6678E1 139D26B7 819F7E90

The base point *G* in compressed form is:

$G =$ 03 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296

Finally the order n of G and the cofactor are:

$n =$ FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551

$h =$ 01

A.4 Recommended Parameters for secp384r1

The elliptic curve domain parameters over $\mathbb{F}_p$ associated with a Koblitz curve `secp384r1` are specified by the sextuple $T = (p, a, b, G, n, h)$ where the finite field $\mathbb{F}_p$ is defined by:

$p =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFF
$= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$

The curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

$a =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFC
$b =$ B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112
0314088F 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF

E was chosen verifiably at random from the seed:

$S =$ A335926A A319A27A 1D00896A 6773A482 7ACDAC73

The base point *G* in compressed form is:

$G =$ 03 AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98
59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7

Finally the order n of G and the cofactor are:

$n =$ FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973
$h =$ 01

A.5 Recommended Parameters for secp521r1

The elliptic curve domain parameters over $\mathbb{F}_p$ associated with a Koblitz curve `secp521r1` are specified by the sextuple $T = (p, a, b, G, n, h)$ where the finite field $\mathbb{F}_p$ is defined by:

$p =$ 01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
$= 2^{521} - 1$

The curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

$$\begin{aligned} a = {} & \texttt{01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF} \\ & \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF} \\ & \texttt{FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFC} \\ b = {} & \texttt{0051 953EB961 8E1C9A1F 929A21A0 B68540EE A2DA725B 99B315F3} \\ & \texttt{B8B48991 8EF109E1 56193951 EC7E937B 1652C0BD 3BB1BF07} \\ & \texttt{3573DF88 3D2C34F1 EF451FD4 6B503F00} \end{aligned}$$

E was chosen verifiably at random from the seed:

$$S = \texttt{D09E8800 291CB853 96CC6717 393284AA A0DA64BA}$$

The base point *G* in compressed form is:

$$\begin{aligned} G = {} & \texttt{03 AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98} \\ & \texttt{59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7} \end{aligned}$$

Finally the order n of G and the cofactor are:

$$\begin{aligned} n = {} & \texttt{01FF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF} \\ & \texttt{FFFFFFFF FFFFFFFA 51868783 BF2F966B 7FCC0148 F709A5D0} \\ & \texttt{3BB5C9B8 899C47AE BB6FB71E 91386409} \\ h = {} & \texttt{01} \end{aligned}$$

Appendix B

Activity Diagrams

The figures presented in this appendix contains the activity diagrams mentioned in section 3.5.1

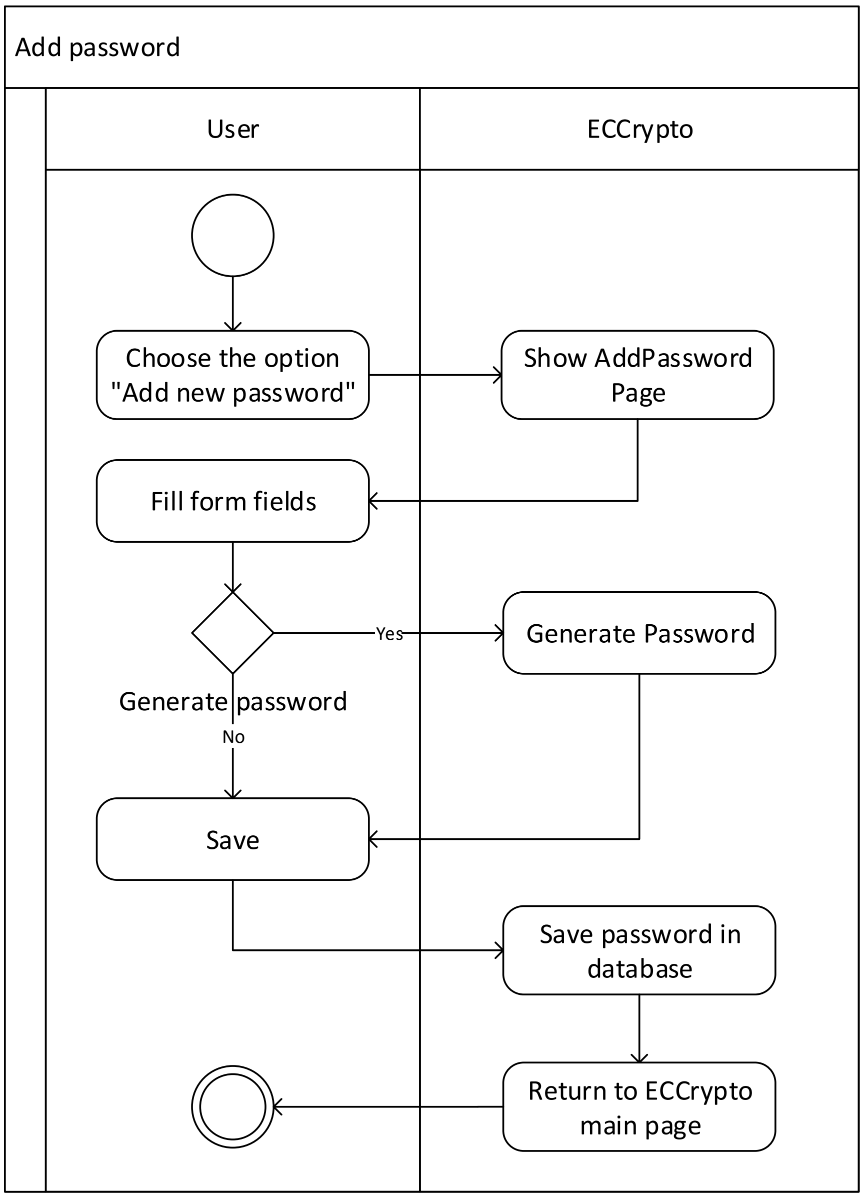


Figure B.1: Diagram for the password activity which depicts the process of adding a new password to `ECCrypto`.

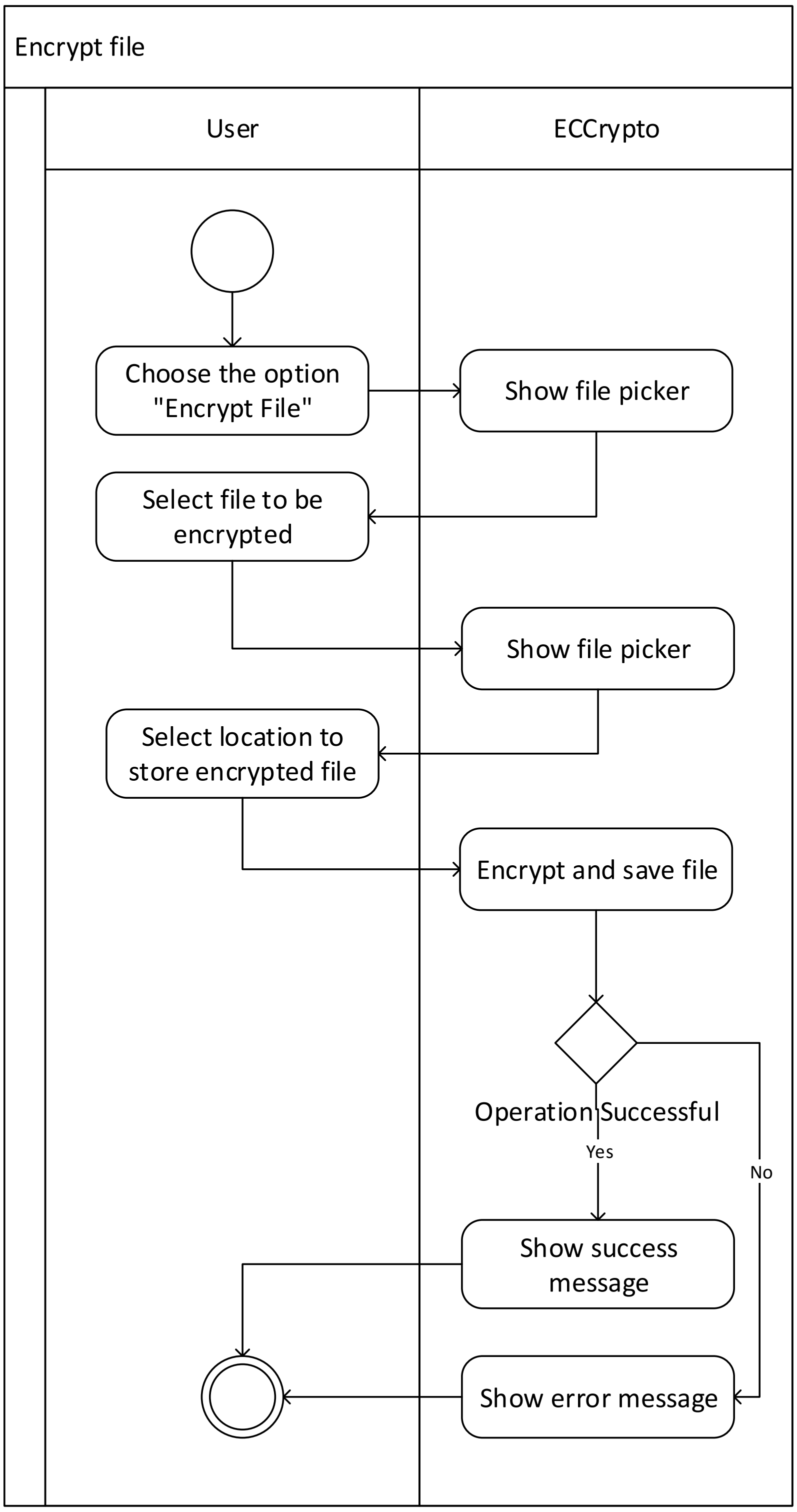


Figure B.2: Diagram for the file encryption activity in which the user chooses a file to be encrypted from the phone and afterwards chooses the location where the encrypted file is to be stored.

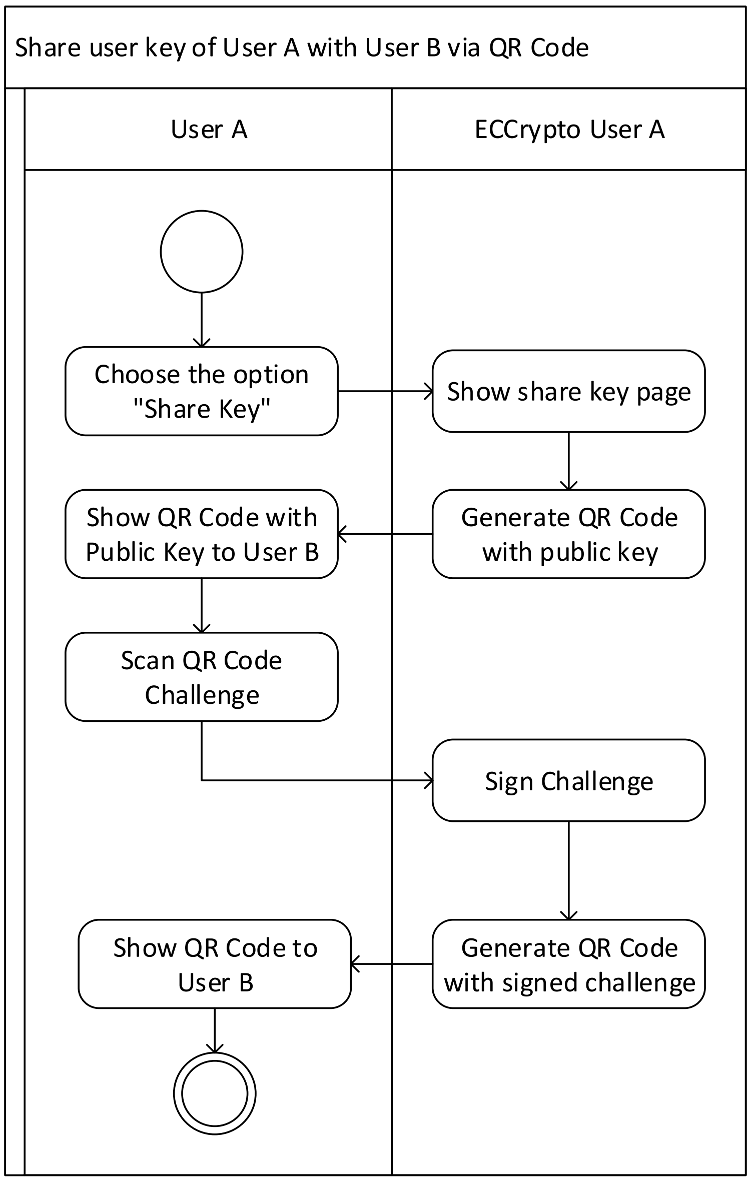

Share user key of User A with User B via QR Code
User A
ECCrypto User A
Choose the option "Share Key"
Show share key page
Generate QR Code with public key
Show QR Code with Public Key to User B
Scan QR Code Challenge
Sign Challenge
Generate QR Code with signed challenge
Show QR Code to User B

Figure B.3: Diagram for the share key activity that shows the process of sharing a key with a contact via QR code.

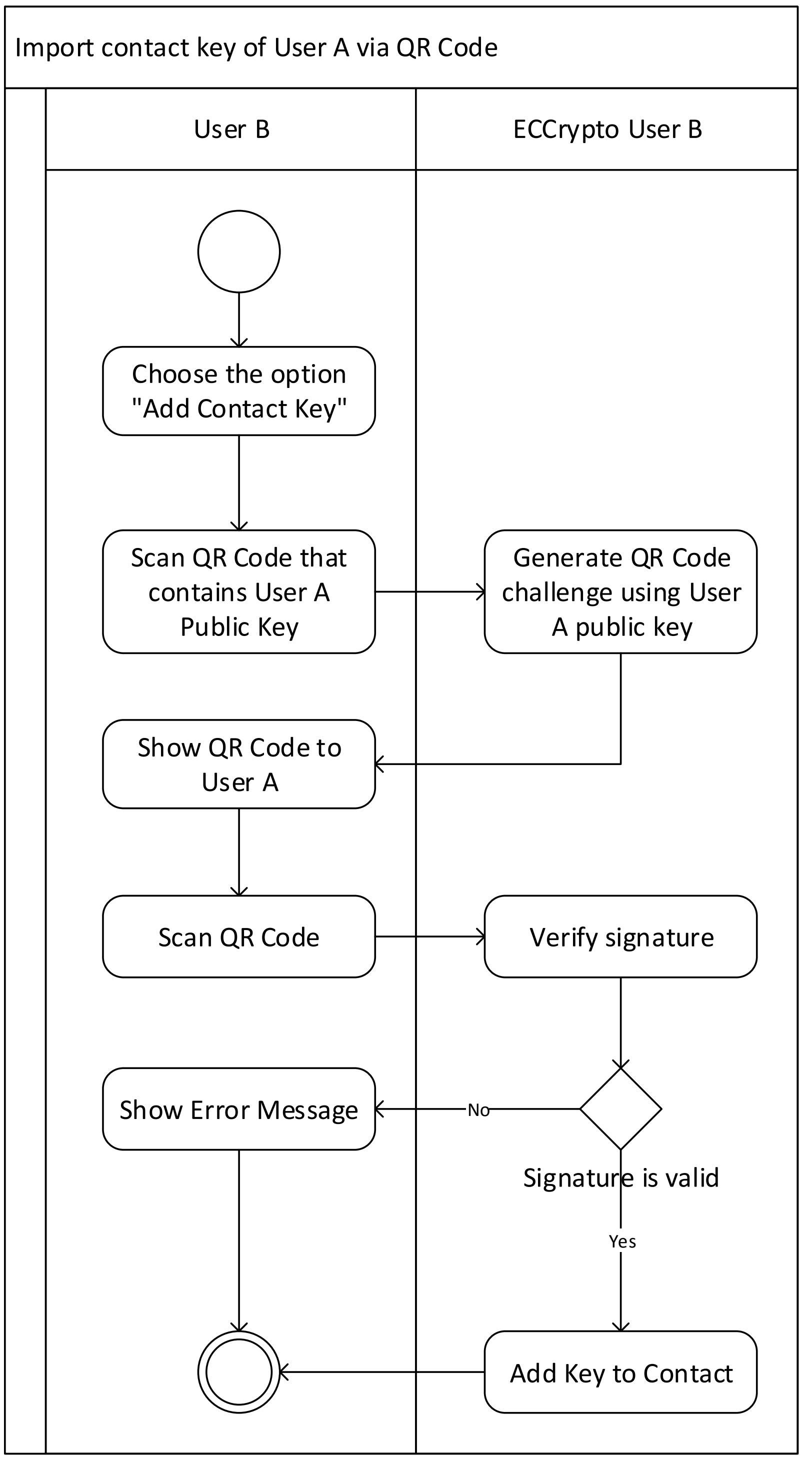

Import contact key of User A via QR Code
User B
ECCrypto User B
Choose the option "Add Contact Key"
Scan QR Code that contains User A Public Key
Generate QR Code challenge using User A public key
Show QR Code to User A
Scan QR Code
Verify signature
Show Error Message
No
Signature is valid
Yes
Add Key to Contact

Figure B.4: Import key activity diagram that shows the process of importing the key of a contact via QR code.