ONLINE SOCIAL NETWORKS AND PRIVACY – LEVERAGING CRYPTOGRAPHY TO GAIN CONTROL

By

JOHANN REHBERGER

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

26/11/2011

ABSTRACT

ONLINE SOCIAL NETWORKS AND PRIVACY – LEVERAGING CRYPTOGRAPHY TO GAIN CONTROL

By

JOHANN REHBERGER

Online Social Networking platforms allow their customers to store and share information. A user might want to store information for themselves or share data with a small group of people, like family and close friends, or potentially everyone in the world.

Users store personal information in online social networks and hence privacy aspects become relevant. If a user wants to keep certain information private or limited to a small audience, the user should be able to control that.

If an attacker compromises a social networking account, the attacker is able to read messages and post new information on the user's behalf. In such a scenario, the attacker is impersonating the user and the user has no privacy control over his or her personal information.

Furthermore, a user may not trust how a service provider protects their data and may therefore prefer not to store information in clear text with the social network. If the clear text data is leaked, anyone can read it, regardless of whether the leak happened accidentally or through vulnerability in the service.

Encryption and signing techniques can enable advanced use scenarios for online social networking platforms to secure content. Social networks provide programmatic interfaces to query and post information. This opens up the avenue for social network client applications that encrypt and sign data before submitting such data to the social network.

The main goal of this project is to research the privacy aspects of social networking platforms (in particular Facebook) and provide mitigation techniques that will help end users protect their data beyond what social networks currently provide.

In particular, a mobile phone application, codenamed “Project Cryptbook” is discussed and implemented that allows users to encrypt information before posting the information to the social network. Even if the data accidentally becomes accessible to a third party, the third party will not be able to decrypt such data without the correct decryption key. Additionally the prototype allows seamless decryption of the encrypted information to provide access to the intended recipient.

DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

JOHANN REHBERGER

Student, Supervisors and Classes:

Student name:	Johann Rehberger
Student ID number:	15633625
GDI name:	Alfred Vella
RMT (GDI) class ID:	ComputingReserachMethodsTraining.2011.04.28.202
DA name:	Yongge Wang
DST (DA) class ID:	ComputingAdvisorClass.20100701.221

ACKNOWLEDGEMENTS

I would like to thank Yongge Wang for his guidance and assistance as supervisor of this dissertation. In addition my thanks and appreciation goes to the University of Liverpool, Laureate Online Education and the students I had the chance to interact and learn from as part of this Master's program.

During the research of this project I had the chance to read a lot of interesting and exciting material that helped me see things in different ways. Therefore, I would like to thank the people who authored those documents and shared their research. I also want to thank Vicky for her help.

Last but not least I would like to say thank you to Maria, Leopold, Hubert and my parents.

Sincerely,

Johann Rehberger.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter 1. INTRODUCTION

1.1 Scope

This project researches the state of online social networking platforms in regards to privacy with a focus on Facebook. In addition it discusses existing and, in the context of social networks and mobile phones, novel mitigation techniques that enable users to take control and prevent unauthorized users from accessing their social networking data.

A mobile application, codenamed “Project Cryptbook” is implemented that leverages client side encryption capabilities before posting information to the social network. This will prevent the social networking platform or other users, who do not possess the appropriate decryption key, to recover the plaintext message.

In addition to encryption, Project Cryptbook enables signing capabilities for posts, which further improves the security of the user accounts and the integrity of messaging on the social network. If an account gets compromised, the attacker is not able to post on a user’s behalf because the victim’s private signing key is not accessible from within the social network.

The mobile application is implemented for Windows Phone 7.

1.2 Problem Statement

Every day millions of users post information on social networking applications. The data is uploaded in clear text. This means that the online social network provider and others can read the content. Additionally, if there is a security vulnerability that allows an attacker to compromise an account or an attacker correctly guesses the user’s password, the attacker can act on the victim’s behalf, reading all information posted and also post information as the victim.

1.3 Approach

- Research the history and current state of social networks (in particular Facebook) from a privacy standpoint
- Identify and research existing privacy protection mechanism of Facebook and third party solutions
- Explore and compare symmetric and asymmetric encryption technologies and how they can be leveraged to improve privacy in online social networks
- Implementation of a mobile prototype application to encrypt and sign messages before posting messages to Facebook. Additionally the application will decrypt the data when viewing it
- The mobile application will leverage OAuth2 to authenticate with Facebook and post, and read cipher text as the consenting user
- Explore side channel attacks
- Explore the usability of a mobile client based encryption solution and the challenges of effective key management
- Investigate and propose further improvements to the implemented solution

1.4 Outcome

In particular the contribution of this project is to build a privacy layer on top of a social network platform (Facebook) that will allow customers to encrypt and sign data before posting it to the social networking site. This solution will put privacy control into the hand of the end user, as they can decide with whom and how to share the encryption keys. The application is implemented for Windows Phone 7 and Facebook.

Chapter 2. BACKGROUND AND REVIEW OF LITERATURE

2.1 Related Work

The earliest paper to discuss Facebook's privacy in detail appears to be by Harvey and Soltrem (2005) and points out that "users may submit their data without being aware that it may be shared with advertisers. Third parties may build a database of Facebook data to sell. Intruders may steal passwords, or entire databases, from Facebook."

As part of their research they conducted a survey with the results that 23% of the respondents were not concerned with Facebook's privacy and 35.5% are barely concerned.

Cavalli et al. (n.d.) conducted a study that reflects the tremendous growth of Facebook. They refer to the Facebook boom and points out that in 2008, 7.9% percent of the students used Facebook, and a year later it was 59.4%.

At the time of this writing the popularity of Facebook has further increased and the privacy concerns have shifted. Governments and private organizations, especially in Europe are investigating the privacy practices of Facebook.

As an example consider the following press release: "The Data Protection Commissioner's Office (Independent Centre for Privacy Protection - ULD) calls on all institutions in the federal state of Schleswig-Holstein, Germany to shut down their fan pages on Facebook and remove social plug-ins such as the "like"-button from their websites. After a thorough legal and technical analysis ULD comes to the conclusion that such features are in violation of the German Telemedia Act (TMG) and of the Federal Data Protection Act (BDSG), respectively the Data Protection Act of Schleswig-Holstein (LDSG SH)."

Independent Centre for Privacy Protection of Schleswig-Holstein (2011)

Schrems (2011) established a web page called Europe-v-Facebook.org that raises concerns and awareness about the privacy implications and data governance practices of Facebook. There is a list of complaints that is brought up against Facebook Ireland and the website points out the right an individual has to request access to all the information a company (in this case Facebook) stores about an individual under the European Data Protection Law.

2.1.1 The Privacy Controls of Social Networking Platforms

Baatarjav et al. (2008) introduced a privacy management system to better help users to protect their personal information and they pointed out that third party applications might gain access and harvest data from the social network.

The discussed system is based on a probabilistic approach using a revelation matrix and calculating the recommended privacy settings. This means that the user is locking down the information that will be accessible by others, especially third party application developers.

There are other papers and articles that discuss the privacy settings and how to adjust them in Facebook in detail to enforce better privacy. Null (2009, p. 97) points out some of the common “disaster” scenarios like oversharing information with the boss, or the stalker problem. The article discusses on how to administer and adjust privacy settings accordingly in order to only share information with people relevant to the topic.

As soon as the user provides data to a third party (in this case Facebook) the user has to fully trust Facebook. For instance Facebook might (and did in the past) change their privacy settings or introduce new services, which might reveal unwanted information and violate the privacy of individuals.

As soon as data is uploaded, the online social network might change a setting or introduce a feature which exposes information that was considered private by the publisher.

During the course of this research Google introduced Google+, another service in the social networking landscape. The privacy protection controls are much related to that of Facebook. A user can define so called “Circles” to communicate with. This seems analogous to Facebook lists, although Goggle+ puts the Circles in the center of the user experience, whereas for Facebook lists, although differences exit. Recently Facebook introduced that address this and made the usage of lists and groups easier and the company also added additional improvements (Cox, 2011).

Similarly to Facebook, Google+ does not provide client side encryption capabilities to give end users more control.

2.1.2 Decentralizing Social Networks

To improve the privacy situation in social networks decentralized networks are proposed (Seong et al, 2011). A concrete implementation of such a system, called PrPl, is provided by Cutillo, Molva and Strufe (2009) and they point out that the data could possibly be encrypted.

In addition to the privacy concerns, there are issues with centralized systems as pointed out by Chi et al. (2010, p. 15): "For example, any central entity can easily be a single point of failure, a single target for denial-of-service (DoS) attacks, and also a bottleneck for network performance."

One of the challenges with decentralized platforms is that a new infrastructure has to be created, which does not leverage the existing momentum of online social networking platforms.

2.1.3 Leveraging Cryptography as a Solution for Privacy Concerns

Lucas and Borisov (2008) introduced FlyByNight that provides encryption capabilities for social networks. The paper notes that the “implementation had to address some challenges that result from using JavaScript and the Facebook application platform; in part, our work was helped by using proxy cryptography.”

This means that a third party service is involved, which allows additional attacks and might raise more privacy concerns.

Luo, Xie and Hengartner (2009, p. 27) worked on FaceCloak, which is a system that provides fake information to the social networking platform and holds the real data in encrypted form on a separate web server, not owned by the social network. The solution is implemented via a Firefox extension and works with Facebook. The drawback is that this system involves another third party that has to be trusted. This model is similar to uProtectIt from the company Reputation, which is discussed below in the Industry Sources section.

NOYB is a system that splits pieces of information in so called atoms and encrypts them on Facebook and it is implemented via a Firefox plugin (Guha, Tang and Francis, 2008). The system does not seem to provide the ability to generate new encrypted content and lacks key management. Guha, Tang and Francis (2008, p. 53) point out that "At present, our implementation does not manage keys and instead defers to the user at the time of encryption and decryption to enter the password, however, it is possible to modify our plugin to automatically distribute keys to the friends, and receive keys from friends through web based email services."

2.2 Literature

Current research frequently highlights ways for users to gain better control of their privacy settings in Facebook by addressing the privacy challenges from within Facebook by locking down the sharing and privacy settings more restrictively (Baatarjav et al., 2008 and Null, 2009). The Privacy Rights Clearinghouse points out that the data might be collected for legal purposes, like advertising or by third party developers, and also illegally, for instance by an identity thief or other criminal activities (Anonymous, 2010).

Hofmann (2007) reports that there was a security incident at Facebook where the source code of the social networking platform was available for download due to a misconfiguration of a server.

Furthermore, Facebook tried to introduce new features like Beacon that caused a lot of public attention because of its automatic opt-in functionality that exposed ones shopping behavior to other users (Perez, 2007).

Facebook changed the privacy and sharing functionality of its product multiple times, which led to confusion, privacy concerns, and unintended exposure of personal information. The companies CEO, Mark Zuckerberg officially apologized for a privacy related incidence in the past, stating that “We really messed this one up”, in regards to the launch of News and Mini-Feeds (Zuckerberg, 2006).

Fowler and Lawton (2011) reported about privacy concerns of US and European law makers around the recent introduction of Facebook’s face recognition and auto-tagging feature.

In order for end users to have better control of their data, a more flexible system seems to be desirable. If users want to keep certain data confidential, the information should be encrypted before the data is made available to a social network.

2.3 Industry Sources

Companies, like reputation.com, are researching in the privacy space and they provide a solution that stores data in encrypted form on their servers instead of Facebook’s (Reputation, 2011). Unfortunately detailed information on how the encryption is performed could not be found. This raises questions on how the encryption keys are managed, what encryption algorithm is used and where and how they store the data.

The idea of this approach is to store the data with a different company in the cloud. Instead of Facebook the user has to trust two companies. Although it has to be pointed out that it creates a level of diversification, and hence provides a slightly increased level of privacy.

Diaspora (2011) is working on a decentralized online social networking platform. This will be a new distributed social network, considering privacy aspects as top priority, and is not built on top of Facebook with its big user base which might make adoption difficult.

The goal of this research is to highlight that an end user who is concerned about the privacy of their information should provide data in encrypted form to social networks.

Project Cryptbook, a prototype application for Windows Phone 7, is implemented to demonstrate the feasibility.

The author of this document agrees with the statement in Facebook's Rights and Responsibilities (Facebook, 2011), that states "Your privacy is very important to us.", and seeks ways to further improve the situation for end users.

2.3.1 Third Party Applications

Facebook allows third party application developers to programmatically gain access to the data of individuals with their consent. It is pointed out in Facebook's Statement of Rights and Responsibilities that "When you use an application, your content and information is shared with the application. We require applications to respect your privacy, and your agreement with that application will control how the application can use, store, and transfer that content and information" (Facebook, 2011).

A user has to grant an application permission to act on their behalf in order to read, post or update information. In fact there is a more granular permission model employed by Facebook.

This expands the scope of the initial trust decision a user made with Facebook. Hence it is advised that user's install applications cautiously.

Chapter 3. THEORY

3.1 Privacy

The Cambridge Online Dictionary (2011) defines privacy as “someone's right to keep their personal matters and relationships secret.” This definition highlights that privacy is a right, and an individual should be able to assert that right.

3.2 Online Social Networking Sites

Online social networking sites existed for a long time, initially applications like ICQ and Messenger enabled users to connect with others and share information, and in the mid-1990s the first generation of online social networking platforms appeared and disappeared again (e.g. sixdegrees.com) These sites might have been too early to achieve critical mass; after all the Internet was not as widespread as it was a decade later (Boyd and Ellison, 2007).

Today social networking sites have hundreds of millions of users (Facebook Statistics, 2011).

3.3 Cryptography

Merriam-Webster Online Dictionary (n.d.) defines Cryptography as “secret writing”. The idea of secretly communicating has a long standing history. Pass and Shelat (2007, p. 4) point out that the first known crypto algorithm, was the so called Ceasar Cipher. The basic idea is to replace each letter in a message by a different one in the alphabet.

Nowadays cryptography is a science which according to Menezes, Oorschot and Vanstone (1996, p. 4) has the following security goals:

- Confidentiality

  Data is kept secret from unprivileged users. When Bob and Alice want to secretly

exchange a message, Eve should not be able to read it. The names Bob and Alice are used commonly throughout the cryptography literature (Rivest, Shamir and Adleman, 1978); and these personas are used for consistency to readers.

This means that cryptography can be used to enforce privacy.

- Data integrity

  Integrity ensures that data has not been altered. When Bob receives a message from Alice he wants to make sure that the message has not been tampered with.

- Authentication

  The goal of authentication is to ensure that the data originates from a particular service or user. When Bob receives a message from Alice, he wants to make sure that it indeed originated from Alice and not someone else.

- Non-repudiation

  In case a dispute arises, non-repudiation allows understanding if a certain action took place or not.

According to Menezes, Oorschot and Vanstone (1996, p. 4) there are two encryption/decryption schemes, namely:

- **Private Key Encryption**

  In this scenario a single secret key is used to encrypt and decrypt information. The secret key has to be known by both entities. Throughout this document this method is called Symmetric Encryption

- **Public Key Encryption**

  Rivest, Shamir, Adleman (1978) propose the encryption scheme with two encryption keys, a private and a public key. The public key is shared publically for eve-

ryone to see. Alice can use the public key of Bob to encrypt a message for Bob. Afterwards Bob can use his private key to decrypt the message from Alice. The security of this scheme depends on the difficulty in factoring large prime numbers. Furthermore, it enables to “sign” messages.

This scheme is referred to as Asymmetric Encryption in this document.

3.4 Facebook and Mobile Usage

Over the last years mobile phones with rich user interfaces and fast internet connections have become reality and many people use their phones to access online social networks. According to Facebook Statistics (2011) “More than 350 million active users currently access Facebook through their mobile devices.”

3.5 Privacy in Social Networks, Cryptography and Mobile Devices

As pointed out, cryptography can be leveraged to achieve better privacy controls for social networking sites. Together with the power of mobile devices the encryption can be performed by client devices that offer such rich capabilities.

Additionally, cryptography can also provide Integrity, Authenticity and Non-Repudiation for social networking platforms.

Chapter 4. ANALYSIS AND DESIGN

In order to design and implement a client application that integrates with Facebook and provides encryption/decryption and signing capabilities, the following minimum features are required:

- Login to Facebook using OAuth2 protocol handshake
- Provide base classes that interact with the Facebook Graph API
- Display the wall and friends of the current user using the Graph API
- Decrypt wall posts that the user is capable of decrypting because the user possesses the correct decryption key
- Allow the user to post to his wall (either in encrypted or unencrypted form)
- Display the Friends of the current user and allow to view, post to the friends wall considering the encryption/decryption features
- Basic key management (create a key, publish the key, simple backup, …)

4.1.1 Basic High Level Application Flow

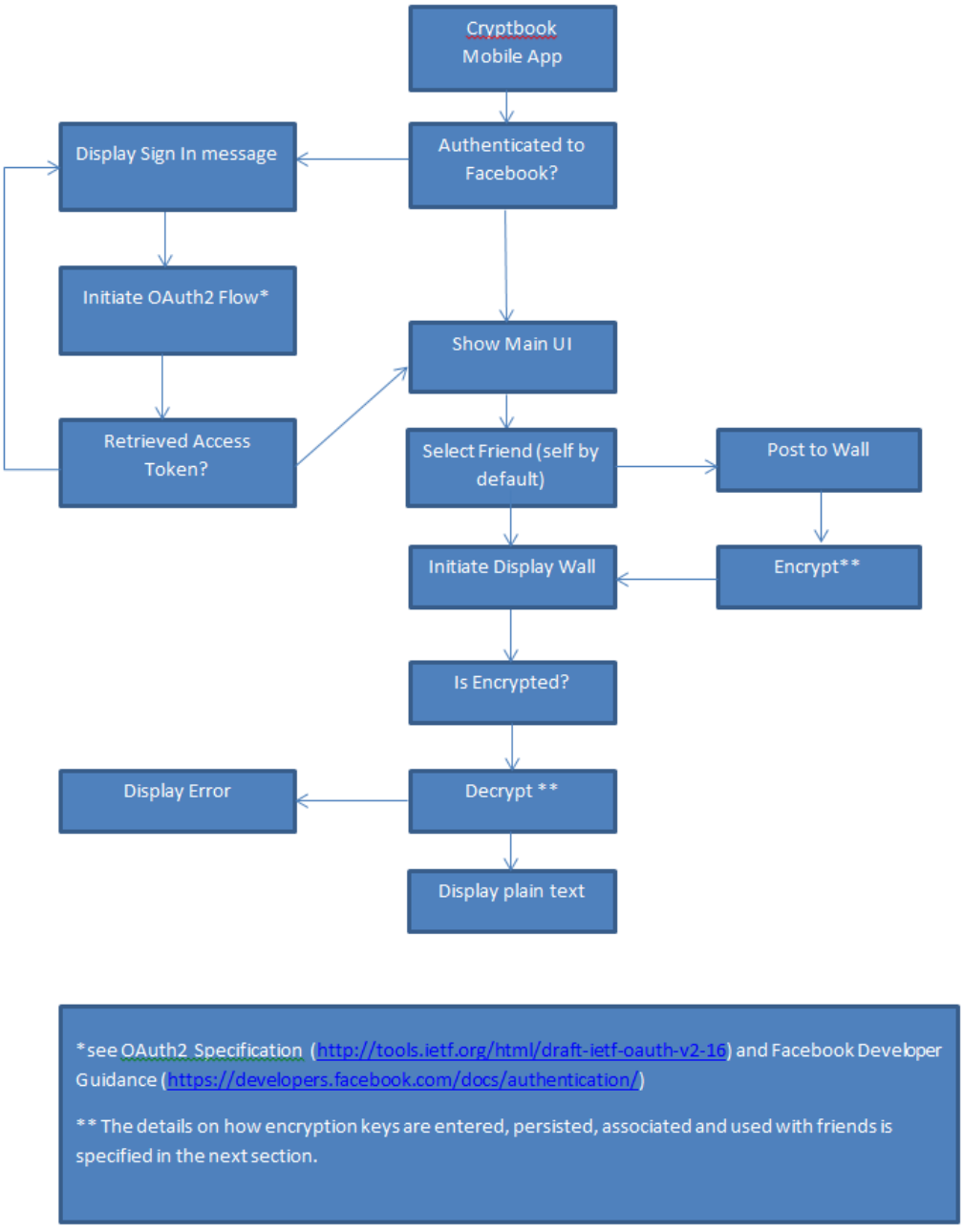


Figure 1 Basic Application Flow

4.2 Facebook Graph API

Facebook has a variety of ways to enables third party applications to access and act on a Facebook user's behalf. The main API is the so called Graph API. The Graph API is leveraging OAuth2 as authentication and authorization technology Graph API (2011).

This means that a call to the Graph API needs to have an access token that is established at the login phase. To integrate on that level, a third party application registers their application with Facebook.

4.2.1 JSON

The responses from Facebook are in the JSON format, additionally xml is available for some APIs. Below is an example response that shows information returned from a Graph API call to retrieve information of a particular wall post:


```
"data": [
      {
         "id": "667753271_10150353910838272",
         "from": {
            "name": "Thomas Kern",
            "id": "667753271"
         },
         "message": "this is a test wall post for Liverpool",
         "actions": [
            {
               "name": "Comment",
               "link":
"http://www.facebook.com/667753271/posts/10150353910838272"
            }…
```


Figure 2 Partial JSON response of a wall post

The data contains information about who posted the information, time and additional privacy information. The privacy field is of interest, as it shows which groups or individuals do have access to the post.

In addition the client has to be able to process the JSON data format. From a design perspective this means that the client application has to be able to provide support for JSON, or a custom parser will have to be implemented. Microsoft .NET and Silverlight for Windows Phone 7 offer the capability to process JSON via the JSON serialization and deserialization classes (MSDN, 2011).

4.3 Client Side Encryption

The proposed system requires client side capabilities that enable encryption, decryption and signing of data with current algorithms, including AES, RSA and SHA256. In addition RFC 2898 has to be supported to enable key generation via a passphrase.

The choice to implement the solution for mobile phones was one of the prerequisites, as the researcher wanted to better understand these systems and their capabilities.

This project explores two approaches, namely symmetric and asymmetric encryption of Facebook data.

4.4 Symmetric Encryption

The initial and simpler approach to perform encryption appeared to be symmetric encryption, where both the sender and recipient share a secret key, derived from a commonly known passphrase with each other. The encryption algorithm of choice is AES.

4.4.1 Creation of the Key

The application will prompt the user for a passphrase and then create the symmetric key according to RFC 2898.

4.4.2 Authenticity

The authenticity of data encrypted with this single symmetric key is still not strong. This becomes especially true if the key is shared with multiple people.

4.4.3 Message Format - Symmetric Encryption

In order to identify messages, successfully decrypt them and enable versioning a message envelope scheme is introduced for both symmetric and asymmetric encryption.

```
#CB|Version|Base64Encoded(IV|AES(Plain Text, IV))
```

Explanation:

- A posted message is split into several segments via the pipe symbol
- The message envelope starts with the phrase #CB which introduces an encryption message
- The version number allows for future upgrades to the message format
- The plaintext is AES encrypted with the Initialization Vector that the application randomly constructs
- Additionally the Initialization Vector is pre-pended to the encrypted message and then the values are base 64 encoded.

Note: It was determined that base 64 encoding alphabet does contain the pipe symbol, the hash tag (#) or the @ sign (Josefsson, 2006). This should prevent various security related injection threats.

The user who posted the message is not included, as this information is part of a Facebook post itself. As will be discussed in the evaluation section, this format has some certain drawbacks.

4.4.4 Key Management for the Symmetric Approach

The challenge with this approach is the exchange of the key. The exchange has to be done out of band. A possible solution to this might be to encrypt the passphrase with another encryption key and send that information as an attachment via email to the other user. This pushes the problem just a level down, as the two users still need to be able to exchange the other passphrase with each other. The symmetric key will be stored on the users phone at all times.

4.5 Asymmetric Encryption

Asymmetric encryption leverages a key pair to perform encryption/decryption and signing. The pair consists of a private and public key. The public key is openly shared by the owner and for instance be published on the users social network account. Whereas the private key stays in the possession of the owner at all times.

Facebook is used to store the user's public key, which enables many useful scenarios. The best location to store the key currently appears to be a Facebook Note. Friends use the public key of a user to encrypt messages for them.

The private key is kept on the phone, although it can be sent via email as a backup. It might also be feasible to store the private key with Facebook, in an encrypted form.

4.5.1 Creation of the Asymmetric Key Pair

The application will create the key pair for the user and allow the user to publish the public key to Facebook as a "Note" with the following format:

Title of the Note:

@Privacy Solution Center - Public Key

Content of the Note:

@CB|Version|KeyID|Base64(Public Key)|FacebookUserID

Description of the message tokens:

Token	**Description**
@CB	Fixed value at the beginning
Version	Current version number of the data structure
KeyID	The unique ID of the public key, represented by a GUID Type 4.

Base64(Public Key)	Base 64 encoded public key. The public key itself is represented in XML format as described in the Microsoft Developer Network (2011):
	`<RSAKeyValue>`
	`  <Modulus>…</Modulus>`
	`  <Exponent>…</Exponent>`
	`  <P>…</P>`
	`  <Q>…</Q>`
	`  <DP>…</DP>`
	`  <DQ>…</DQ>`
	`  <InverseQ>…</InverseQ>`
	`  <D>…</D>`
	`</RSAKeyValue>`

FacebookUserID	The ID of the user on Facebook as described in the Graph API.

Table 1 Public Key Message Format

4.5.2 Authenticity

An improvement to the symmetric encryption is that asymmetric encryption enables the scenario of signing a message with the private key by encrypting a hash of the message with the private key and appending it to the post. The recipient can decrypt the message with the private key, and validate the signature by decrypting the appended signature with the public key of the sender and validating that the resulting hash matches the hash of the decrypted message.

4.5.3 Key Management for the Asymmetric Encryption Approach

In this case a key pair has to be managed. The public key has to be made available to other users in order for them to use it for encrypting messages to the owner of the public key. As pointed out earlier Facebook as a sharing platform seems to be the ideal platform to distribute and share the public key.

4.5.4 Storing the Private Key

The private key, which is used for signing and decryption, has to be kept on the user's phone. To increase security it should encrypted with a passphrase, and requires decryption/unlocking it before use.

4.5.5 Using Facebook as the Backup Location for the Private Key

To further increase the usability of the application, especially considering scenario where a phone crashed and all data is lost or the user buys a new phone. An interesting additional scenario that is explored, although currently not implemented in the prototype is the idea of using Facebook as the storage location of a symmetrically encrypted private key. The application could provide a feature to encrypt the private key with a passphrase and publish it as Facebook "Note". This will enable the recovery of the private key in case the phone is lost or damaged.

The format of the suggested private key backup is similar to publication of the public key, the difference would be in the title of the posted Note including metadata that this is a private key, and that the content of the Note has to include the AES encrypted private key.

Private Key Backup Note:

```
@Privacy Solution Center - Private Key
```

4.5.6 Message Format – Asymmetric Encryption

```
#CB|
Version|
Base64(AES(message, IV))|
Base64(Recipient1RSA(aeskey):cipher1)#Base64(Recipient2(aeskey):cipher2)#…|
Base64(IV)|Base64(Signature)
```

Note: the lines breaks are for clarity only.

Explanation:

Token	**Description**
@CB	Fixed value at the beginning
Version	Current version number of the data structure
message	The plain text message, which will be symmetrically encrypted using AES
IV	Initialization Vector for the symmetric AES encryption
aeskey	The AES key
Base64	Base64 encoding function
RecipientxRSA	The AES symmetric key encrypted with the public key of Recipient x.

cipher*X*	Result of Recipient*x*RSA (the encrypted symmetric key)
Signature	SHA256 hash of the entire message, beginning at #CB up to the final pipe symbol, and encrypting the resulting hash with the private key of the sender.

Table 2 Asymmetric Message Format

Detailed Explanation:

- The message is split into segments using the pipe symbol |
- The message starts with the pre-fix #CB to indicate an encrypted message
- Version: The version number of the message format
- AES (message): The plain text message encrypted with a the AES symmetric key
- The message can be encrypted for 1 to n recipients. The following segment contains the AES key encrypted for each recipient. The keys' are separated with the pound sign #.
- The Initialization Vector the message was symmetrically encrypted with
- Signature: The signature is computed by hashing the entire message block beginning at #CB up to the final pipe symbol, and encrypting the resulting hash with the private key of the sender.

4.6 Additional Considerations and Threats

4.6.1 Message Length

The resulting message, including base 64 encoding might become long, compared to ordinary Facebook wall posts. Initial investigations showed that Facebook wall posts were limited to a set of 400 characters at some point in the past, although this limitation does not exist anymore, as the application is capable of posting messages that are much longer.

4.6.2 Messages considered as Spam

Regular users of Facebook might get confused with the posts of the application because they are not able to read it and it could be considered spam to them. This could have a variety of impacts:

- Users reporting the application as spam
- Users become more interested and involved in data privacy and the ability to encrypt information
- Social networks realizing the potential of client side privacy controls and adopting such solutions
- Tighter integration with Facebook, by allowing a feature that only a specific application can read the data it posted itself. A user viewing the wall through the through the browser can choose not to display them.

A solution could be for social networking platforms to adopt the ideas described in this document and make client side encryption capabilities a first class citizen of social networking applications and tightly integrating it, this would be the best way to mitigate the spam concerns.

4.6.3 Key Rotation and Re-Encryption

An area that will need additional thought and has not been explored in details is the long term key management including potentially re-encrypting already existing information in case a key has stolen and a user wants to change the encryption key.

4.6.4 Side channel attacks

The proposed solution is still leaking a lot of information about who communicates with whom and what the length of their messages is. This means that there are still many information pieces that can be inferred from the data, even though the exact content is unknown to someone who does not possess the decryption key.

4.6.5 Padding Oracle

Signature validation has to happen before any decryption operation takes place otherwise an application might expose a decryption oracle (Paterson et al, 2004). Since the decryption happens locally within the mobile application this threat seems low severity, as there is no direct attack vector, compared to a web service that provides decryption capabilities.

4.6.6 Replay Attacks

It was discovered that the messaging format is vulnerable to a replay attack, in order to prevent such an attack it will be useful to include a timestamp in the message format in a future revision. The timestamp can be compared to the post date and should be within given tolerance ratio.

4.6.7 Account compromise and overwriting the public key

In case an attacker compromises the account of a Facebook user, he will not be able to decrypt the existing content that was previously encrypted. Although the attacker could publish a new public key as Note on the victim's Facebook profile and then intercept all future communication.

Chapter 5. METHODS AND REALIZATION

5.1 Facebook Server Side Application Registration

The codename of the project is ***Project Cryptbook***. Interestingly, registering the server side part of the application with the name Cryptbook was rejected by Facebook, containing the statement: “Our system has detected that App Name contains a disallowed use of the term “book”.

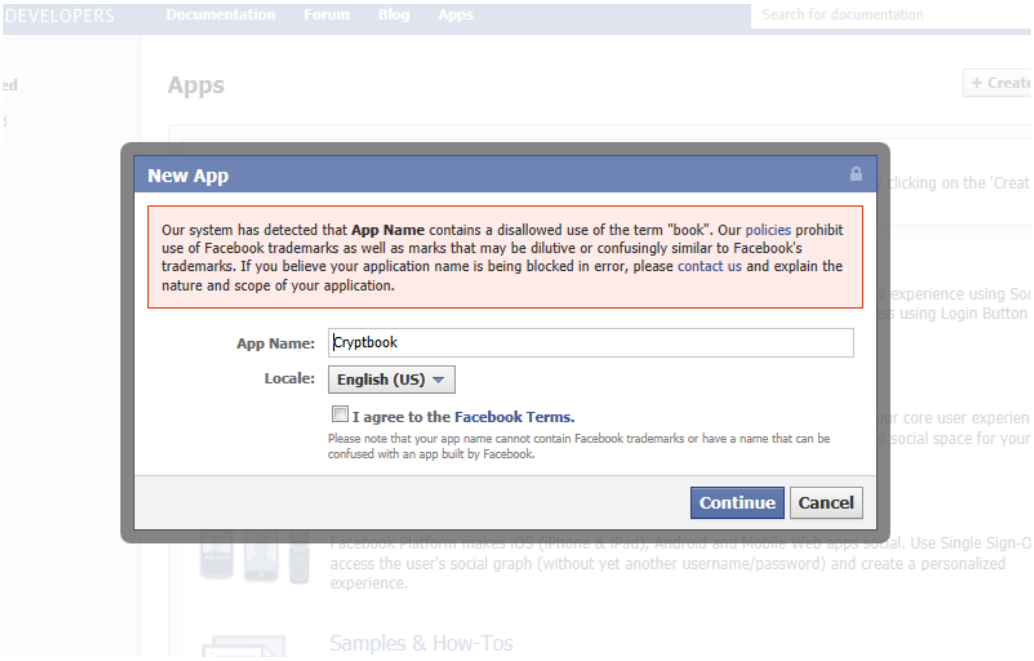


Figure 3 Facebook Application Registration

Apparently Facebook prevents server side third party Facebook applications to use the term *book* in the application name. Hence the application was registered at Facebook with the name “Privacy Solution Center (Alpha)”.

5.2 Windows Phone Application – Project Cryptbook

The initial prototype is implemented for Windows Phone 7.5 (also referred to as Mango). The software to develop applications (Visual Studio Express, incl. the emulator) can be downloaded for free from the Microsoft website.

The Windows Phone 7 API's provide encryption capabilities for both asymmetric and symmetric encryption. Additionally signing capabilities (e.g. SHA 256) are available.

Deploying custom code

In order to deploy custom code to a physical phone, one has to register as a Windows Phone 7 developer with Microsoft and unlock the phone. Through Microsoft DreamSparks student's program, this is possible without additional costs for the student.

5.3 Silverlight and the User Interface

A big piece of the research for this project was to become familiar with Windows Phone 7 and with the Facebook Graph API. This document will discuss the source code related to authentication, encryption/decryption and key generation, and slightly touch Facebook's Graph API. The user interface code will not be discussed in detail. The entire relevant source code can be found in the Appendix.

5.4 Authentication

The core of the OAuth2 authentication is done via hosting a web browser control that displays the OAuth2 consent flow.

From an implementation point of view the most important piece is the *LoadCompleted* event of the web browser control, which will be invoked multiple times during the OAuth2 handshake with Facebook and ultimately retrieves the Access Token necessary for the authenticated Graph API operations.

```
private void webBrowser1_LoadCompleted(object sender,
                            System.Windows.Navigation.NavigationEventArgs e)
{
   //inital load?
   if (!(e.Uri.AbsoluteUri.ToLower().StartsWith(FacebookHelper.RedirectURL))
      && (!(e.Uri.AbsoluteUri.ToLower() == "about:blank")))
   {
      webBrowser1.Visibility = System.Windows.Visibility.Visible;
      StopPleaseWaitAnimation();
   }

   //Access Token retrieved?
   if (e.Uri.AbsoluteUri.ToLower().StartsWith(FacebookHelper.RedirectURL))
   {
     App.AccessToken = e.Uri.OriginalString.Substring(
     e.Uri.AbsoluteUri.IndexOf("#access_token=") + 14);

     if (App.AccessToken != string.Empty)
     {
        App.IsUserAuthenticated = true;
        webBrowser1.Navigate(new Uri("about:blank"));
        webBrowser1.Visibility = System.Windows.Visibility.Collapsed;

        StartPleaseWaitAnimation();

        LoadCurrentUser();
     }
   }
}
```

5.5 JSON Serialization and Deserialization

In order to enable JSON serialization/deserialization with the .Net Framework classes, the data structures are decorated with [DataContract] and [DataMember] attributes. Below is an example of the Item data structure that builds the foundation of other structures like WallPost, and Friend are:

```
namespace Cryptbook.DataContracts
{
    [DataContract]
    public class Item : BaseItem
    {
        private string id;

        [DataMember(Name = "id")]
        public string Id
        {
            get { return id ?? ""; }
            set { id = value; NotifyPropertyChanged("id"); }
        }
    }
}
```

5.6 Symmetric Encryption Prototype

The first prototype leverages AES symmetric encryption. The symmetric encryption key is derived from a passphrase according to RFC 2898 (Kaliski, 2000).

Below is the helper method that implements the symmetric encryption capabilities:

```
/// <summary>
/// Encrypts the given plain text using a passphrase and given IV.
/// The key generation is according to RFC 2898.
/// </summary>
/// <param name="passphrase">The passphrase to create the symmetric key</param>
/// <param name="data">The plain text to encrypt</param>
/// <param name="initializationVector">The salt for the key generation</param>
/// <returns>encrypted data</returns>
public static string Encrypt(string passphrase, string data, string initializa-
tionVector)
{
    Rfc2898DeriveBytes key = new Rfc2898DeriveBytes(passphrase,
                Encoding.Unicode.GetBytes(initializationVector),
                1000);

    AesManaged aes = new AesManaged();
    aes.Key = key.GetBytes(aes.KeySize / 8);
    aes.IV = Convert.FromBase64String(initializationVector);

    byte[] cipherTextBuffer;
    using (MemoryStream memoryStream = new MemoryStream())
    {
       using (CryptoStream cryptoStream = new CryptoStream(memoryStream,
       aes.CreateEncryptor(), CryptoStreamMode.Write))
        {
           byte[] clearTextBuffer = Encoding.Unicode.GetBytes(data);
           cryptoStream.Write(clearTextBuffer, 0, clearTextBuffer.Length);
           cryptoStream.FlushFinalBlock();
        }
        cipherTextBuffer = memoryStream.ToArray();
    }

    return Convert.ToBase64String(cipherTextBuffer);
}
```

To improve the computation costs for an attacker to brute force passphrases, RFC 2898 (Kaliski, 2000) **Error! Reference source not found.Error! Reference source not found.**recommends a minimum of 1000 iterations.

Screenshot of the initial prototype using symmetric encryption:

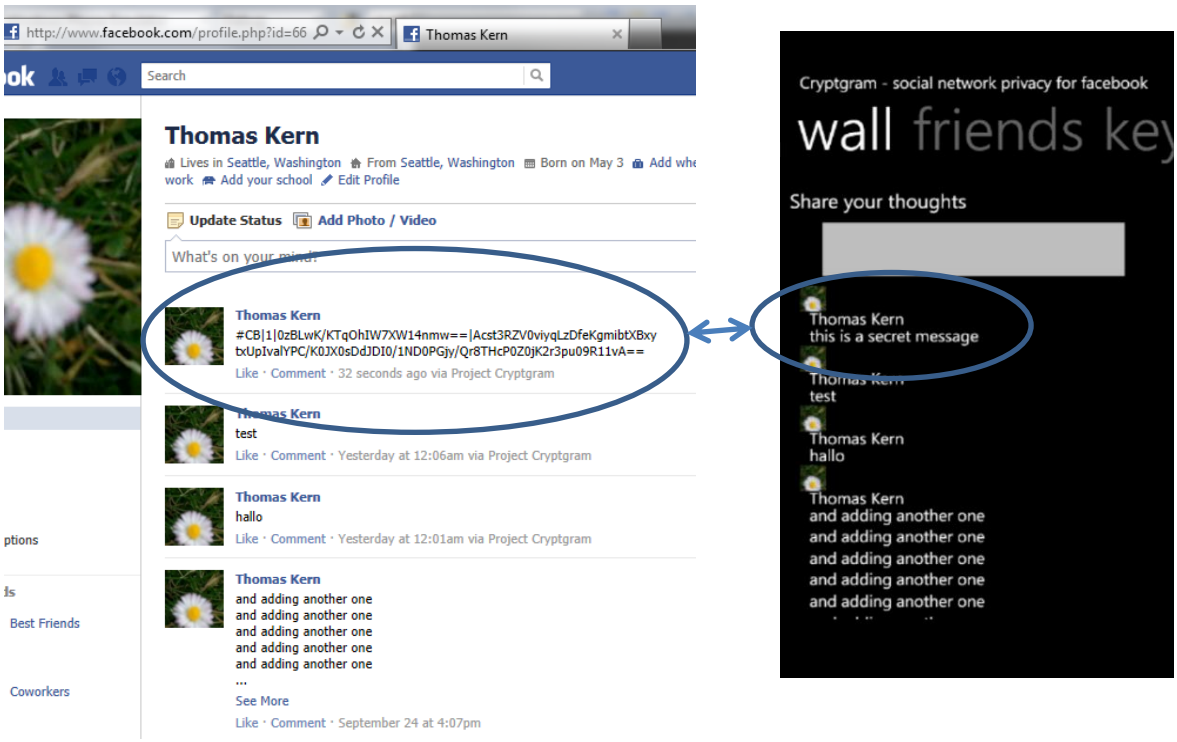


Figure 4 Early protoype using symmetric encryption

5.6.1 Realizations after implementing the early Prototype

After finishing the early prototype it became clear that symmetric encryption, although it initially appeared to be simpler, has a set of disadvantages that make it challenging for users to adopt. In particular:

- **Entering the passphrase on the phone**

  It is a cumbersome and lengthy procedure to enter a long and secure passphrase on a mobile device. This will likely lead to users choosing very simple passphrases like "*password*" or "*test",* and hence can easily be broken.

Figure 5 Passphrase

- **Sharing of the symmetric key**

  The “email symmetric key” features, although it appeared to be straight forward requires the end users to copy and paste the symmetric key from the phone’s email application, which requires at least 6 clicks for a user to achieve:

  1) Open the mail message
  2) Select the symmetric key
  3) Click the copy icon
  4) Navigate to the target application
  5) Identify the correct place/tab within the target application
  6) Paste the symmetric key into the target application

  The above list is a best estimate; many users will require more navigation steps.

- **Lack of signing**

  The symmetric key does not provide the way to know who encrypted the message. This means that anyone in possession of the symmetric encryption key can claim or refute to have encrypted a certain message.

  The authentication mechanism of Facebook is the only mitigating factor. In case a user account is hijacked an attacker could easily reply existing encrypted messages.

- **Discovering which key was used to encrypt a certain message**

  To display the wall of a user, the application crawls through each message to identify if it requires decryption. This is a challenge, as the application has to identify which key was used to encrypt the message in order to successfully decrypt it. As message structure does not possess that information, nor is there a

central lookup available for all keys, the application is brute forcing each key to see if it decrypts the post successfully. This approach is slow and non-ideal.

The author considered and identified the following solutions to this problem:

1) Create a hash of the key and store the hash together with each message
2) Before decrypting the message, check if the user possesses the key by validating the hashes (incl. a salt value)

Unfortunately it was not entirely clear if this opens up a “rainbow table attack” or chosen plain text attack against the hashes, as the salt used for the encryption would likely have to be the same across many posts.

The mentioned drawbacks, most importantly the usability issues, caused the shift and focus towards the realization of an asymmetric solution. Nevertheless the source code of the symmetric encryption remains and can be found in the Appendix.

5.7 Asymmetric Encryption Prototype

The implementation of the asymmetric encryption prototype addresses some of the drawbacks of the symmetric approach, and the main benefit is the increased usability of the solution.

Creation and Storage of the Asymmetric Key Pair

The .NET framework has the `RSACryptoServiceProvider` class that is used to create the key, encrypt and decrypt data. The application has a data structure called `AsymmetricKey` that holds additional metadata information about the key. The key creation code is as follows:

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(2048);
AsymmetricKey            key = new AsymmetricKey();

key.Id        = Guid.NewGuid().ToString();
key.User      = App.CurrentUser.Id;
key.IssueDate = DateTime.UtcNow.ToString("s");
```

```
    key.PrivatePublicKey = rsa.ToXmlString(true);
    key.PublicKey        = rsa.ToXmlString(false);
```

A current weakness of this approach is that the key material is stored in clear text in the process memory and also in the Isolated Storage of Silverlight of the mobile application, and that is an accepted risk of the prototype.

After they key pair is created the user can publish the Public Key to Facebook as a Note, which is done via the following method:

```
   public void PostNote(string facebookId, string subject, string message,
                        Action<UploadStringCompletedEventArgs> callback)
   {
     string wallPost =
          string.Format("https://graph.facebook.com/{0}/notes?access_token={1}",
                  facebookId,
                  accessToken);

     Uri uri = new Uri(wallPost);

     string postBody = string.Format("subject={0}&message={1}",
                              HttpUtility.UrlEncode(subject),
                              HttpUtility.UrlEncode(message));

     Post(uri, postBody, callback);
}
```

The message format of an encrypted post was described in the previous section. Below is the C# method signature that creates an appropriately structured message:

```
public static string EncryptAsymmetric(List<AsymmetricKey> publicKeyMaterial,
                                       AsymmetricKey       privateKeyMaterial,
                                       string              message)
```

The method takes the following arguments as input: a list of public keys (the recipients of the message), the private key of the sender, which is used for signing and the plain text message. The decryption code is structured similarly and the entire source code can be found in the Appendix.

Figure 6 Screenshot of Project Cryptbook

Chapter 6. RESULTS AND EVALUATION

The application consists of the following deliverables:

- **Facebook Application:** Privacy Solution Center (Alpha)
- **Windows Phone 7 client application**: Cryptbook – the Windows Phone 7 client application for the Privacy Solution Center
- **Documentation**

6.1 Project Cryptbook - A Client for the Privacy Solution Center

Figure 7 Main Menu of Project Cryptbook

Figure 8 Project Cryptbook view compared to browser view of Facebook

6.2 Emulator and Physical Device

The application was tested on both an emulator and a physical device.

The availability of both allows for more efficient concurrency testing, where on the emulator User A is logged in and the physical device was used by User B. Then they could send and receive messages from each other.

6.3 Test cases

6.3.1 Setup and Configuration

The application was tested with 3 Facebook user accounts:

- Bob Cipher
- Alice Crypto
- Johann Rehberger

Bob and Alice are friends on Facebook. Bob is also friends with Johann. Alice and Johann are not friends.

6.3.2 Usability

The initial prototype used symmetric encryption and was discarded quickly after performing the usability tests, where it was realized that a mobile device is not a good option. Below are the pros and cons of the symmetric encryption.

Advantages of Symmetric Key

- **Group messages**

  The usage of a shared symmetric key works fine with a group of well-known users that can exchange the key (or passphrase for that matter) out of band. The length of the encrypted messages is typically smaller, compared to the asymmetric encryption, especially if there are many recipients involved.

Disadvantages of Symmetric Key Encryption

- **Strength of the Passphrase**

  The encryption key is derived from a passphrase that the user enters. The longer a passphrase is, the more difficult it becomes for an attacker to guess the individual characters (McDowell, Rafail and Henan, 2004). Unfortunately on the phone it

is not as simple to enter long phrases for password and it adds a lot of complexity in using the product. Users might end up with simple passphrases that will be easy to guess.

- **Key Exchange**

  In order to confidentially exchange messages on Facebook, the passphrase has to be shared between the individuals that would like to leverage it for encryption. There is a set of options on how this could be performed, although to provide a more secure key exchange the communication channel has to be a different one. An example would be that Alice and Bob use the phone to communicate the passphrase to each other, and afterwards enter it on the phone and going leverage the resulting symmetric encryption key for further confidential communication on Facebook. Another option would be to send the passphrase in clear text via email, although an attacker might sniff the clear text passphrase.

  If the passphrase is transferred via an email, the users could leverage the Copy & Paste functionality to import the passphrase to the application. This solution would somewhat mitigate the usability concerns described in the first bullet point.

- **Lack of Signing**

  The messages are not authenticated beyond what Facebook provides with the username/password protection.

Benefits of the Asymmetric Encryption and Signing

- **Out of Box Encryption and Signing Capabilities**

  As soon as Bob has published his public key as a Facebook Note, Alice can start posting encrypted messages for him. There is no out of band key exchange required which streamlines the process of confidential communication.

  The solution does not require to remember a passphrase, or to persist one anywhere.

Drawback of the Asymmetric Encryption and Signing

- **Long Term Key Management**

  Although not entirely unique to the public/private key approach the long term key management is challenging, and a solution for key rotation and re-encryption of data with a new key needs to be researched and provided. Otherwise the solution will have usability challenges long term, e.g. if a key pair is compromised or the user wants to start using a new key pair.

- **Backup of the Private Key**

  The user has to back up the private key securely in order to be able to decrypt and sign messages. If the private key is lost, the data will not be recoverable. A potential solution to this problem is to also upload the private key to Facebook (potentially encrypted with a passphrase).

The mentioned usability drawbacks of the symmetric encryption, especially the requirement to exchange the passphrase out of band, appeared to be a big usability problem, compared to the asymmetric key pair solution that requires minimum setup and configuration effort.

6.3.3 Privacy & Security

Since the symmetric solution was discarded the focus of the testing is for the asymmetric solution. Below is the description of the test cases that were used to validate the functionality.

Test Case 1: Bob logs on to Cryptbook

Test Case 1.1: Bob authorizes Cryptbook to perform operations on his behalf

Test Case 2: Alice logs on to Cryptbook

Test Case 2.1: Alice authorizes Cryptbook to perform operations on her behalf

Test Case 2: Bob views his wall

Test Case 3: Bob views his friend list

Test Case 4: Bob generates his public/private key pair

Test Case 5: Bob publishes the public key to Facebook as a Note

Test Case 6: Bob posts a message on Alice's wall

Test Case 7: Bob posts the message "this is a confidential message for Alice only" (encrypted for Alice and Bob) and signed by Bob on Alice's wall

Note: Expected result is that the operation fails, as Alice does not yet have a private key.

Test Case 8: Alice generates his public/private key pair

Test Case 8.1: Alice publishes the public key to Facebook as a Note

Test Case 9: Bob posts the message "this is a confidential message for Alice only" (encrypted for Alice and Bob) and signed by Bob on Alice's wall

Test Case 10: Bob posts the comment "this comment can be read by anyone" to the message posted in Test Case 9 in clear text

Test Case 11: Bob posts the comment "this is an encrypted comment" (encrypted for Alice and Bob) on Alice's wall

Test Case 12: Bob views Alice's wall and decrypts all the messages and comments

Test Case 13: Alice views her wall and the application decrypts the messages for her.

Test Case 14: Alice posts a message on her wall (only encrypted for her)

Test Case 14.1: Bob cannot decrypt the message on Alice' wall created in Test Case 14

Test Case 15: Johann views Bob's and Alice's wall in the browser and sees the encrypted messages, not the plain text ones

6.3.4 Performance

Performance was not a consideration during the implementation of the prototype. The application does not implement a form of caching to improve performance. Testing showed that the computing power and connection speed of the mobile device, using 3G, is sufficient to handle the amount of requests the application issues when loading the wall and pictures of users.

The computing required to decrypt messages is noticeable very limited in the UI. The application does not load more than 20 wall posts (excluding comments) from Facebook at a time, which keeps the decryption effort limited.

Chapter 7. CONCLUSIONS

During the research, design and implementation of this project the author had the chance to learn a lot about application development for mobile devices, in particular Windows Phone.

7.1 Lessons Learned

Initially it seemed more manageable to leverage symmetric encryption to achieve the goal of this project. Although it turned out that asymmetric encryption is more powerful and useable.

The asymmetric approach does not require to secretly exchanging the key out of band, or entering the key on the phone. A phone's input capabilities are limited and the pass-phrase for key generation should possess a certain minimum length and character complexity in order to be strong. This is the core limitation, as in my opinion most users will choose a passphrase similar to "password", "test", etc…

With asymmetric encryption the secure out of band exchange of a key is not necessary. This provides an advantage, which is that as soon as the user publishes the public key to the social network, anyone can access it and is able to securely exchange messages with that user.

7.2 Future Activity

The opportunities that cryptography enables to achieve social network privacy are tremendous. Further research and the creation of an international standard that specifies how client side encryption and social networks work is proposed.

7.3 Prospects for Further Work

In order to enable the client side encryption capabilities of the “Privacy Solution Center” for more platforms a version for Android and iPhone should be implemented. Furthermore a rich Desktop client application would be another possibility for future work in this area.

The feature set of the tool can be expanded to include encryption and decryption of pictures and other Facebook data elements.

Facebook as Public Key Distributor

The idea of using Facebook as the distributer of public keys should be explored further as this enables many useful scenarios. The current work is focused on communicating with mobile applications and wall posts, although there are many more scenario where the publication of the user’s public key on Facebook can be leveraged to communicate in a confidential and authentic way with that user.

PGP Integration

PGP is commonly used by privacy minded people. The prospect to integrate Project Cryptbook and PGP would further advance the understanding, on how client side encryption can help users to protect their information in online social networks. This might especially be useful in conjunction with a desktop application.

REFRENCES CITED

Anonymous (2010) Fact Sheet 35: Social Networking Privacy: How to be Safe, Secure and Social. Privacy Rights Clearing House / UCAN. [Online] Available from: http://www.privacyrights.org/social-networking-privacy#access (Accessed 11th May 2011)

Baatarjav E.-A., Dantu, R., Phithakkitnukoon, S. (2008) Privacy Management for Facebook [Online] Available from: http://dx.doi.org/10.1007/978-3-540-89862-7_23 (Accessed 11th May 2011)

Boyd, D. M., Ellison, N. B. (2007) Social Network Sites: Definition, History, and Scholarship. University of California-Berkeley. Michigan State University. [Online] Available at: http://jcmc.indiana.edu/vol13/issue1/boyd.ellison.html (Accessed 29th October 2011)

Cambridge Online Dictionary (2011) Definition of Privacy [Online] Available from: http://dictionary.cambridge.org/dictionary/british/privacy (Accessed 29th October 2011)

Cavalli, N., Costa, E. I., Ferri, P., Mangiatordi, A., Micheli, M., Pozzali, A., Scenini, F., Serenelli, F. (n.d) Facebook influence on university students' media habits: qualitative results from a field research. *University of Milan-Bicocca*. [Online] Available at: http://web.mit.edu/comm-forum/mit7/papers/MangiatordiMicheliScenini_Facebook_influence_on_university_students.pdf (Accessed 22nd October 2011)

Chi Z., Sun, J., Zhu, X., Fang, Y. (2010) Privacy and security for online social networks: challenges and opportunities. University of Florida and Xidian University. *IEEE.* [Online] Available from: http://dx.doi.org/10.1109/MNET.2010.5510913 (Accessed 2nd November 2011)

Cox, C. (2011) Making it easier to share with who you want. *Official Facebook Blog* [Online] Available from: https://blog.facebook.com/blog.php?post=10150251867797131 (Accessed 23rd October 2011)

Cutillo, L.A., Molva, R. Strufe, T. (2009) Safebook: A privacy-preserving online social network leveraging on real-life trust. Communications Magazine, IEEE, vol.47, no.12, pp.94-101. DOI: 10.1109/MCOM.2009.5350374 [Online] Available from: http://dx.doi.org/10.1109/CSE.2009.387 (Accessed 2nd November 2011)

Diaspora (2011) Share what you want, with whom you want. [Online] Available from: https://joindiaspora.com/ (Accessed 17th May 2011)

Facebook (2011) Statement of Rights and Responsibilities [Online] Available from: http://www.facebook.com/terms.php. *Facebook.* (Accessed 17th May 2011)

Facebook Statistics (2011) Statistics [Online] Available from: http://www.facebook.com/press/info.php?statistics (Accessed, 2nd November 2011)

Fowler, G. A., Lawton, C. (2011) Facebook again in Spotlight on Privacy. *Wall Street Journal* [Online] Available from: http://online.wsj.com/article/SB10001424052702304778304576373730948200592.html (Accessed 25th June 2011)

Graph API (2011) Facebook Graph API. *Facebook.* [Online] Available from: https://developers.*facebook*.com/docs*/api* (Accessed 22nd October 2011)

Guha, S., Tang, K., Francis, P. (2008) NOYB: Privacy in Online Social Network. *Cornell University, Ithaca.* [Online] http://dx.doi.org/10.1145/1397735.1397747 (Accessed 2nd November 2011)

Harvey, J., Soltrem, J. H. (2005) Facebook: Threats to Privacy [Online] http://ocw.cput.ac.za/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-805Fall-2005/8EE6D1CB-A269-434E-BEF9-D5C4B4C67895/0/facebook.pdf (Accessed 22nd October 2011)

Hofmann, H. (2007) Facebook's source code goes public. *CNET* [Online] Available from: http://news.cnet.com/8301-13515_3-9758702-26.html (Accessed 25th June 2011)

Independent Centre for Privacy Protection for Schleswig-Holstein (2011) ULD to website owners: „Deactivate Facebook web analytics". *ULD*. [Online] https://www.datenschutzzentrum.de/presse/20110819-facebook-en.htm (Accessed 22nd October 2011)

Josefsson, S. (2006) RFC 4648: The Base16, Base32, and Base64 Data Encodings. Base 64 Alphabet. *IETF.* [Online] http://tools.ietf.org/html/rfc4648#page-5 (Accessed 22nd October 2011)

Kaliski, B. (2000) PKCS #5: Password-Based Cryptography Specification Version 2.0 RFC 2898. *RSA Laboratories* [Online] Available from: http://www.ietf.org/rfc/rfc2898.txt (Accessed 22nd October 2011)

Lucas, M. M., Borisov, N. (2008) FlyByNight: mitigating the privacy risks of social networking. *University of Illinois at Urbana-Champaign. ACM.* [Online] Available from: http://doi.acm.org/10.1145/1456403.1456405 (Accessed November 2nd 2011)

Luo, W., Xie, Q., Hengartner,U. (2009) FaceCloak: An Architecture for User Privacy on Social Networking Sites. *IEEE.*, p. 26-33. [Online] Available from: http://dx.doi.org/10.1109/CSE.2009.387 (Accessed 2nd November 2011)

Microsoft Developer Network (2011) RSA.ToXmlString [Online] Available from: http://msdn.microsoft.com/en-us/library/system.security.cryptography.rsa.toxmlstring.aspx (Accessed 22nd October 2011)

McDowell, M., Rafail, J., Hernan, S. (2004) Choosing and Protecting Passwords. *Carnegie Mellon University.* [Online] Available from: http://www.us-cert.gov/cas/tips/ST04-002.html (Accessed 31st October 2011)

Menezes, A. Oorschot, P., Vanstone, S. (1996) Handbook of Applied Cryptography, Chapter 1. *CRC Press.* [Online] Available from: http://www.cacr.math.uwaterloo.ca/hac/about/chap1.pdf (Accessed 29th October 2011)

Merriam-Webster Online Dictionary (n.d.) Definition of Cryptography [Online] Available from: http://www.merriam-webster.com/dictionary/cryptography secret writing (Accessed 29th October 2011)

MSDN (2011) JSON Serialization. *Microsoft Developer Network.* [Online] http://msdn.microsoft.com/en-us/library/bb410770.aspx (Accessed 22nd October 2011)

Null, C. (2009) How to avoid Facebook & Twitter Disasters. *PC World.* August 2009.

Pass, R., Shelat, A. (2007) A course in Cryptography. [Online] Available from: http://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf (Accessed 29th October 2011)

Paterson, K. G., Yau, A. K. L (2004) Padding Oracle Attacks on the ISO CBC Mode Encryption Standard. *University of London.* [Online] Available from: http://dx.doi.org/10.1007%2f978-3-540-24660-2_24 (Accessed 2nd November 2011)

Perez, C. J. (2007) Facebook's Beacon More Intrusive Than Previously Thought. *PC World* [Online] Available from: http://www.pcworld.com/article/140182/facebooks_beacon_more_intrusive_than_previously_thought.html (Accessed 25th June 2011)

Rivest, R. L., Shamir, A., Adleman, L. (1978) A Method for obtaining Digital Signatures and Public-Key Cryptosystems. *MIT Laboratory for Computer Science and Department of Mathematics.* [Online] Available from: http://people.csail.mit.edu/rivest/RivestShamirAdleman-AMethodForObtainingDigitalSignaturesAndPublicKeyCryptosystems.pdf (Accessed 29th October 2011)

Reputation (2011) uProtect.it – Take Back Control From Facebook [Online] Available from: http://uprotect.it/index (Accessed 17th May 2011)

Schrems, M. (2011) Europe vs. Facebook [Online] Available from: http://europe-v-facebook.org/EN/en.html (Accessed 22nd October 2011)

Seong, S.W., Seo, J., Nasielski, M., Sengupta, D., Hangal, S., Teh, S. T., Chu, R., Dodson, B., Lam, M. S. (2010) PrPl: a decentralized social networking infrastructure. *ACM.* [Online] Available from: http://dx.doi.org/10.1145/1810931.1810939 (Accessed 2nd November 2011)

Zuckerberg, M. (2006) An Open Letter from Mark Zuckerberg. *Facebook Blog* [Online] Available from: http://blog.facebook.com/blog.php?post=2208562130 (Accessed 25th June 2011)

APPENDICES

Appendix A. APPLICATION FLOW

A.1 Logon and User Consent

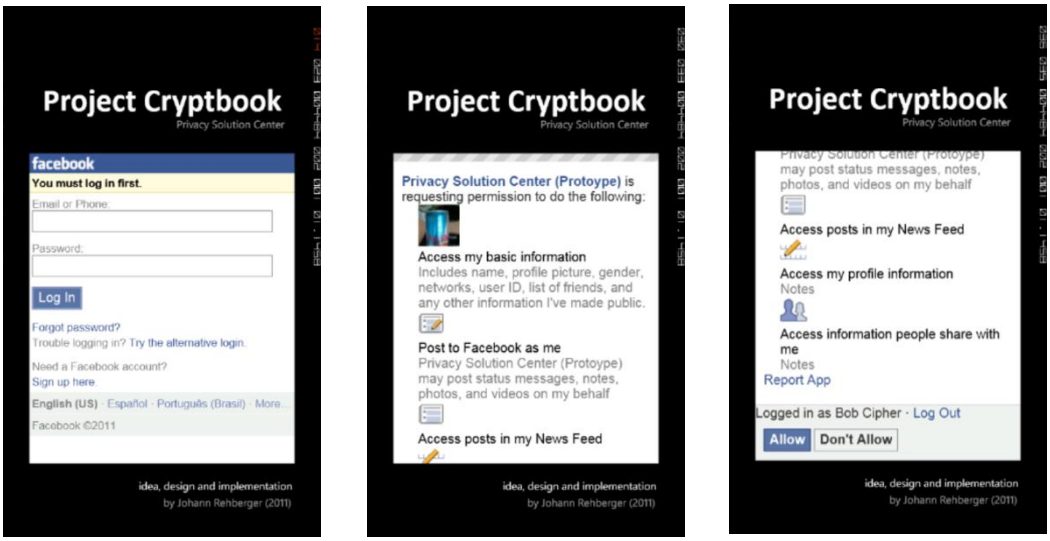


Figure 9 Initial logon and user consent to use the Privacy Solution Center

A.2 Viewing the Wall and Friends

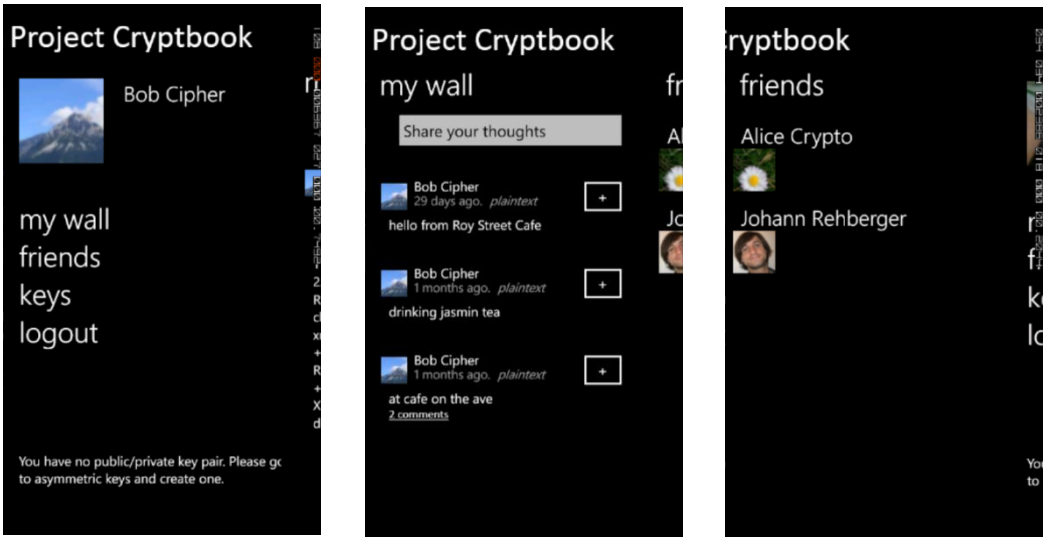


Figure 10 Main Menu, the Wall and viewing Friends

A.3 Generating the public/private key pair

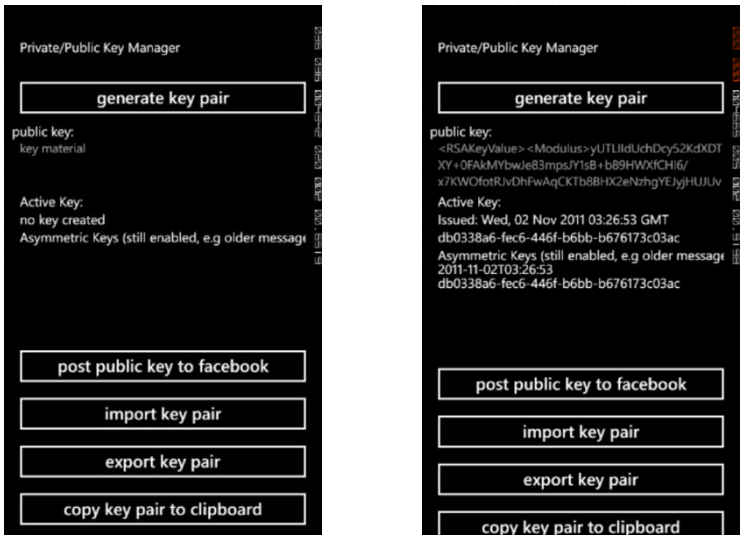


Figure 11 Key pair generation

A.4 Publishing the public key to Facebook as a Note

Figure 12 Publishing the public key as a Facebook Note, and viewing it

A.5 Bob encrypts a message for himself and posts it on his wall

Figure 13 Encrypting a message and viewing it in Cryptbook and in the browser

A.6 Bob encrypts a message for Alice

Figure 14 Encryption UI flow

A.7 View of a wall that contains encrypted and plaintext messages and comments

This demonstrates the view of a full encrypted conversion as seen in Project Cryptbook and in the browser view of Facebook.

Figure 15 Viewing a full conversation in Project Cryptbook compared to browser

A.8 Bob doesn't possess the decryption key for Alice's private posts

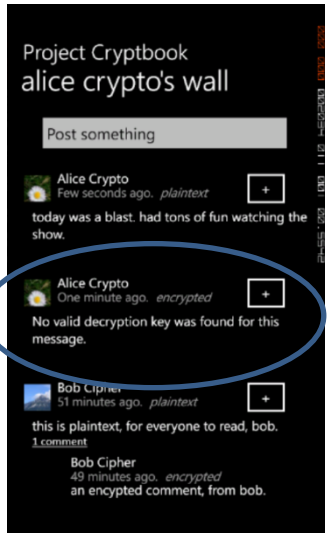


Figure 16 No valid decryption key

A.9 Symmetric Encryption – Key Generation and Sharing Options

The early prototype leveraged symmetric encryption, below are two screenshots that show how the key generation and sharing of the key was implemented.

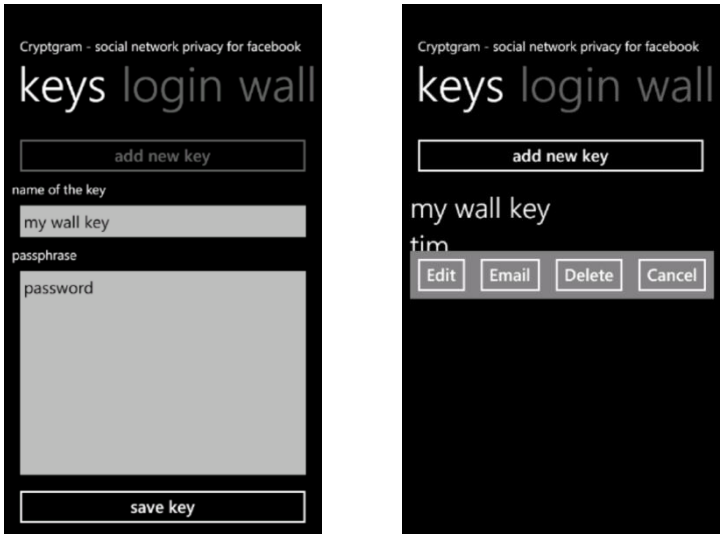


Figure 17 Creation of the symmetric key via a passphrase

Appendix B. Example Posts From Project Cryptbook

B.1 Bob's Public Key posted as a Facebook Note

@CB|1|db0338a6-fec6-446f-b6bb-
b676173c03ac|%3cRSAKeyValue%3e%3cModulus%3eyUTLIIdUchDcy52KdXDTXY%2b0FAkMYbwJe83mpsJ
Y1sB%2bb89HWXfCHl6%2fx7KWOfotRJvDhFwAqCKTb8BHX2eNzhgYEJyjHUJUvbqli%2bzz%2bWvrPd225U0
Q%2bzYwjH1ehtDTa1wxlXeaKey%2bpUCAHl9rDMGK762CNLBdtNIFYgfVDCWP7M7dI2WKgjHGIMZnACl7Tz
F79rR05TKAfkM%2fKkuHMsK%2fjAXBQkt7%2fBB4Fsxn8wZS%2bhiGXlza3rePiVSr6fqjjeM1t1xNQfpIyf0NACw
UmJiHYX9HirXUL9%2b5nqfh7fqJpLmqgRGQQq5QjSc8hQYCyKO3e4ihkI1wuh3lw5a9Lw%3d%3d%3c%2fMod
ulus%3e%3cExponent%3eAQAB%3c%2fExponent%3e%3c%2fRSAKeyValue%3e|100002977350787

B.2 Bob's Private Key represented in XML

<RSAKeyValue><Modulus>yUTLIIdUchDcy52KdXDTXY+0FAkMYbwJe83mpsJY1sB+b89HWXfCHl6/x7KWOfo
tRJvDhFwAqCKTb8BHX2eNzhgYEJyjHUJUvbqli+zz+WvrPd225U0Q+zYwjH1ehtDTa1wxlXeaKey+pUCAHl9rD
MGK762CNLBdtNIFYgfVDCWP7M7dI2WKgjHGIMZnACl7TzF79rR05TKAfkM/KkuHMsK/jAXBQkt7/BB4Fsxn8w
ZS+hiGXlza3rePiVSr6fqjjeM1t1xNQfpIyf0NACwUmJiHYX9HirXUL9+5nqfh7fqJpLmqgRGQQq5QjSc8hQYCyKO
3e4ihkI1wuh3lw5a9Lw==</Modulus><Exponent>AQAB</Exponent><P>9iban7hUnHaHHxQtiz1BjK+7iNquD19c
B5c1WAM49pox6qCJJNMHiA8orqIOSnqNCaW8IUfimes6Ie8IxEKO8k/pSk9nGIvkhN1bin2cShGqwELx79EJMz
WdJG9xa4LuWOH0SBgG23ypCpGv5fEs6vl2EDykXWiQpRd1eKNz/2M=</P><Q>0VI8hiiq9etltLtfw1EQg//rDdC
m780bl6BNlmPj8DfoR2+w340k6eZqnj64E4FEGEteiHXoR+VIKw+rox2VsQsqflbrNX3m72meZxrwzj8kQwEj/WD
FrvMorukMpKlUBJU9MchcvOjrDB7mhIEdM/s9WGi2bHo6GyXLExcc0sU=</Q><DP>ifZk+rhZ1Agq26R3kl8W6j9
koOPQQxVzg12grF/CAZWS/KpVs6oDwqKcbqSt1diyTAH0W8LqotJQlsxe4YFclTtsyhWO9xcRUdavhAO7IVidl3z
uFTfYLDeUh+Necp/D71VVkULGNSa9qArXx81RG7IOFjOm/COrTpzDXoZecXk=</DP><DQ>ArpFm6HwCGBW
5PIEH3Fw+6dcEsQDSOCtWV53ZFxM3hZ8amHkNiWKs5mTDhjXITnv/zkMNE8FBdrRA+6iV+/MwY/Zr+fGDN5lI
f2IWOOa+7UK7eyX4XdfXBZDE3Rp+8NRGvYu4/UucC/A0Fmeu19LWCAP/pTuu58kHrJxYQBOGdk=</DQ><In
verseQ>oE9ZzEUgaIjM+Z/XnCAUbxta5bvvlLRKRK2LjCL2p8YW2qho1vgiAqk7pQQnzMw+X2+Hs1N0xdE9Utb
ws+6kvjOPWm4A1KReVIa76GE4kAuFJNY9dYW7u1k6mnHG/ssyenXYs1IBiaAvzEqIIcPOgfVGQPEK5KBzEFix
cjOI3/I=</InverseQ><D>Ru0mQ+1M1fGZTWvUGjTLV4JHrHPP2DVChk12BbghRHSLWe2KsD2iu78aCE6n4AC
IqU/W47LkTFv65vu0ayt77m/RMzFvzeTNJhs7MYcsJu5f9OpIBKBW5qAfFbMsd4fpqUDQ/GIsgq+6M1P6kjl/6Pvf
cT0M0HKCYzFL0lL+QgvuHNFWFGEIwp5pYjQjqCKIoXZ5/UjDnHjG4H7vNwkcXVOooEKY6nqTttuvOZeBNYua
HxgPAsrOILKmI2H9JTGz2jP8tG94cB0YqJ6Urrx5HPcnxpGU1BV2nwgCfSFJmOHFN7pxoH4OBe2zDEZAsXdi/
xIFRjbbdZFRTn9Wti1OuQ==</D></RSAKeyValue>

B.3 Alice's Public Key posted as a Facebook Note

@CB|1|b8ee21ca-11eb-43b0-af00-
7497827406f4|%3cRSAKeyValue%3e%3cModulus%3esrNdU1cLtpaRsl6LseEaGv6wcBo0Ap64CD3fSAszTBd
QgDdIZPVXcjnXxg6PDn%2fDdzu6xLmUYv9QtiyEVKlviR1gOd%2feiVa1MaXwtQuWGWUs9aZA6BMEx0l6b7P
XA%2b%2fSTUfHQ2qi0QWgw5aBo1mcewvVK5X8jivDJOX7kaqbz8Bef1bv220CZTWniXc6COXSc5NlTSMZX1j
9bMGY2%2fu0VPQb%2bPFm7AQhjUFgtRr03FQ1OXkTmnyuF8NBggOERp8ghMK%2fZTPxvdLlAjzYkr1tk9qQ
UTjgIAyRKUdfvFfTLO98ZNQ2VYTjxcHJ2l3exqXxv0%2bz2Cu%2bmfW%2b4kqgFC9KQQ%3d%3d%3c%2fMod
ulus%3e%3cExponent%3eAQAB%3c%2fExponent%3e%3c%2fRSAKeyValue%3e|667753271

B.4 Alice's Private Key represented in XML

<RSAKeyValue><Modulus>srNdU1cLtpaRsl6LseEaGv6wcBo0Ap64CD3fSAszTBdQgDdIZPVXcjnXxg6PDn/Dd
zu6xLmUYv9QtiyEVKlviR1gOd/eiVa1MaXwtQuWGWUs9aZA6BMEx0l6b7PXA+/STUfHQ2qi0QWgw5aBo1mce
wvVK5X8jivDJOX7kaqbz8Bef1bv220CZTWniXc6COXSc5NlTSMZX1j9bMGY2/u0VPQb+PFm7AQhjUFgtRr03F
Q1OXkTmnyuF8NBggOERp8ghMK/ZTPxvdLlAjzYkr1tk9qQUTjgIAyRKUdfvFfTLO98ZNQ2VYTjxcHJ2l3exqXxv0
+z2Cu+mfW+4kqgFC9KQQ==</Modulus><Exponent>AQAB</Exponent><P>47urJhPq8BSTAM3llYZd25p1iN3
66BdQ9PcZM9JquLxA/jS/1pywiO8Lsmtzc15M/fxoccSEKO0Lg+X7g04WpY0qC6eAOlVe1BK5wClMGkCQz2H8
KagF78xX3gsv95ZAVFjwuSnYVSryawUCkGqpIiiXibU23vbWtnZKH7zFvPM=</P><Q>yOGqf6NLSrcvk97qKmH
0u+q6VvIEpJ0Kdr4YrOC6P9duGWtdzY0/VnEsqoGhUXaK6Yf3wFU43TSrIRXM7PPT5yJzX31cSL6FGCwuqa+y
sioygOu+dXfD0DKkcoNOK6Upaoex9aqpiAj/buOxDp39PGVnfOZc3e7E497y4e4W2Ps=</Q><DP>3OeqeWFQw
Fn0zw7q6URRvzfxUzagsIoNidI3rEEsLM7P7MaWYEPWKr5XBlecwghzftxdJB6QZHKsfOGWPCR/a0+nvW5Zw
DwXpHgVnwVwQPbogYJ27KGEQLxjb4xCl1Q0Sp2LxNGGe/cQ+Cb/+b69GdluRbJ264ldddkAxhCDb2k=</DP><
DQ>k8S01/l1/Dsuq3bSCGVQUx+C/UxAsBq5RMnnetxBBY9idR5uvTMHAX51HeQHl9Y8rzcOv5OxJb+Fluvjd/a
QzTihZ//eb8B+RV7S4I3XRIoX5pAD5YM5LqDiKT5/tUlkYvH4rb9hwCjmRVsGtrrvOotZih4m+DaY7Qrm3d13Pxs=
</DQ><InverseQ>kyU4+iuVvjsyVQZCVIRSUQle24PIDSyTvSNQYRr0ABvAZLpTXpOR/HurofUR3GabGAQ4ISB
A0ttkCRwBGg8baG/T76aJR9d/aa4usJvns/G+rvfxih6moayvdewgpxHSpwRwlN4UKE5fksRuHcKKnZNM3tEzclw
8g2H9fDTWyvc=</InverseQ><D>UQm2EM0wqqStfc4SW6BRTlyhdrTrUSTZICOu1eT+ZktMXlizo9T+yvYgO9ju
Vwzpf3C+yee+uwY9e9/EQwBzbCK+OIClUnLXyKQP/Io8j1eqg9vZI2U8jYNs2qsbntVq8VDTtGbvXwrGVOwz7ff9
Vm+PRCzIU1DVNi51XyGXdyZrVKpxDOsqqRPeQeJRJ+ScnO7U4mL0r7Y+Fpg+U4bNrl+2EPI3J/w35mRoKpVj
woFEXBPkb5Z6fMJLWTQBaADb6ZT8GBGORdHxTeNEM0eXniLFDNB8ebnTUPKo94n4l/amcwRjBsZwfD9n+J
fUQD1easbDSO9GFpuqZ6z9YZX34Q==</D></RSAKeyValue>

B.5 Alices' public and private key as stored internally


```
<?xml version="1.0"?>
<AsymmetricKey xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <User>667753271</User>
  <Id>b8ee21ca-11eb-43b0-af00-7497827406f4</Id>
<PrivatePublicKey>&lt;RSAKeyValue&gt;&lt;Modulus&gt;srNdU1cLtpaRsl6LseEaGv6wcBo0Ap64CD3fSAszTBd
QgDdIZPVXcjnXxg6PDn/Ddzu6xLmUYv9QtiyEVKlviR1gOd/eiVa1MaXwtQuWGWUs9aZA6BMEx0l6b7PXA+/S
TUfHQ2qi0QWgw5aBo1mcewvVK5X8jivDJOX7kaqbz8Bef1bv220CZTWniXc6COXSc5NlTSMZX1j9bMGY2/u0V
PQb+PFm7AQhjUFgtRr03FQ1OXkTmnyuF8NBggOERp8ghMK/ZTPxvdLlAjzYkr1tk9qQUTjgIAyRKUdfvFfTLO9
8ZNQ2VYTjxcHJ2l3exqXxv0+z2Cu+mfW+4kqgFC9KQQ==&lt;/Modulus&gt;&lt;Exponent&gt;AQAB&lt;/Exponen
t&gt;&lt;P&gt;47urJhPq8BSTAM3llYZd25p1iN366BdQ9PcZM9JquLxA/jS/1pywiO8Lsmtzc15M/fxoccSEKO0Lg+
X7g04WpY0qC6eAOlVe1BK5wClMGkCQz2H8KagF78xX3gsv95ZAVFjwuSnYVSryawUCkGqpIiiXibU23vbWtnZ
KH7zFvPM=&lt;/P&gt;&lt;Q&gt;yOGqf6NLSrcvk97qKmH0u+q6VvIEpJ0Kdr4YrOC6P9duGWtdzY0/VnEsqoGhU
XaK6Yf3wFU43TSrIRXM7PPT5yJzX31cSL6FGCwuqa+ysioygOu+dXfD0DKkcoNOK6Upaoex9aqpiAj/buOxDp3
9PGVnfOZc3e7E497y4e4W2Ps=&lt;/Q&gt;&lt;DP&gt;3OeqeWFQwFn0zw7q6URRvzfxUzagsIoNidI3rEEsLM7P7
MaWYEPWKr5XBlecwghzftxdJB6QZHKsfOGWPCR/a0+nvW5ZwDwXpHgVnwVwQPbogYJ27KGEQLxjb4xCl1
Q0Sp2LxNGGe/cQ+Cb/+b69GdluRbJ264ldddkAxhCDb2k=&lt;/DP&gt;&lt;DQ&gt;k8S01/l1/Dsuq3bSCGVQUx+C
/UxAsBq5RMnnetxBBY9idR5uvTMHAX51HeQHl9Y8rzcOv5OxJb+Fluvjd/aQzTihZ//eb8B+RV7S4I3XRIoX5pAD
5YM5LqDiKT5/tUlkYvH4rb9hwCjmRVsGtrrvOotZih4m+DaY7Qrm3d13Pxs=&lt;/DQ&gt;&lt;InverseQ&gt;kyU4+iu
VvjsyVQZCVIRSUQle24PIDSyTvSNQYRr0ABvAZLpTXpOR/HurofUR3GabGAQ4ISBA0ttkCRwBGg8baG/T76a
JR9d/aa4usJvns/G+rvfxih6moayvdewgpxHSpwRwlN4UKE5fksRuHcKKnZNM3tEzclw8g2H9fDTWyvc=&lt;/Inver
seQ&gt;&lt;D&gt;UQm2EM0wqqStfc4SW6BRTlyhdrTrUSTZICOu1eT+ZktMXlizo9T+yvYgO9juVwzpf3C+yee+uw
Y9e9/EQwBzbCK+OIClUnLXyKQP/Io8j1eqg9vZI2U8jYNs2qsbntVq8VDTtGbvXwrGVOwz7ff9Vm+PRCzIU1DVN
i51XyGXdyZrVKpxDOsqqRPeQeJRJ+ScnO7U4mL0r7Y+Fpg+U4bNrl+2EPI3J/w35mRoKpVjwoFEXBPkb5Z6fM
JLWTQBaADb6ZT8GBGORdHxTeNEM0eXniLFDNB8ebnTUPKo94n4l/amcwRjBsZwfD9n+JfUQD1easbDSO9
GFpuqZ6z9YZX34Q==&lt;/D&gt;&lt;/RSAKeyValue&gt;</PrivatePublicKey>
<PublicKey>&lt;RSAKeyValue&gt;&lt;Modulus&gt;srNdU1cLtpaRsl6LseEaGv6wcBo0Ap64CD3fSAszTBdQgDdI
ZPVXcjnXxg6PDn/Ddzu6xLmUYv9QtiyEVKlviR1gOd/eiVa1MaXwtQuWGWUs9aZA6BMEx0l6b7PXA+/STUfHQ
2qi0QWgw5aBo1mcewvVK5X8jivDJOX7kaqbz8Bef1bv220CZTWniXc6COXSc5NlTSMZX1j9bMGY2/u0VPQb+
PFm7AQhjUFgtRr03FQ1OXkTmnyuF8NBggOERp8ghMK/ZTPxvdLlAjzYkr1tk9qQUTjgIAyRKUdfvFfTLO98ZNQ
2VYTjxcHJ2l3exqXxv0+z2Cu+mfW+4kqgFC9KQQ==&lt;/Modulus&gt;&lt;Exponent&gt;AQAB&lt;/Exponent&gt;&
lt;/RSAKeyValue&gt;</PublicKey>
  <IssueDate>2011-10-30T23:29:11</IssueDate>
</AsymmetricKey>
```

B.6 Bob posted on Alice's wall: "this is for alice – top secret".

The encrypted string looks as follows (encrypted with Alice's and Bob's public Key and signed with Bob's private key:

#CB|1|zZi325z4ukAjt8Eov+Sg84tmftnaIy1mMNhmq8A5B0uxIbkQc+/G6hWJaruwWI57jTsZKLLDY1L/+XOqHbMAbw==|db0338a6-fec6-446f-b6bb-b676173c03ac:i9eVVyAAkj7mHNGNuP43fxXEap7Yxbcoo1Xi/IkjVWGslJCHETGJCz0xQkHL/z4iQ7OH8DKqeEFu+omKLf3s8+KG4JPYswhqT38ma+Mu4ZJPxQysWBEwMEeJ6shOBqKOp12DeysN6JHoMyKwApMJqxSk+KqsNWjAS/d+u0D3uwNf3xUarwQLM76Eos991BtcK0MzBDjLYTE+L/slr5OL3GLqSXuIAbRgza29slgaxCs/EEsiRQJxJHk3jrCg9arL9RjhdhqTpONukG4JvtEMIoSqqwoRUnmF0N30wmCosxC8E8VU8Go0X9ft1wyVfh1eO1LQqdL5ATZ2duWUEHw78Q==#b8ee21ca-11eb-43b0-af00-7497827406f4:eJOgFxI+MT/xee/Wibr8gdXs6iVCW6PFEV8eoNSvZAz4ojocP6MuqjUpWbO5wg2aHMPMhpLZVI5aCe3OpIWq/N6TKUmu5ll1HYbQYwWKM1X4rG+sZxtwq+iJ6TbQ+jtebEcom6eCuXOPZ7CeO4/QYWkCEIqG8hY6m0QI+hUM7CvCBP6U0OzgwdgOPPbkUd57eZ9Pf3QtDc+OrVat1cFgDxwaCqcSLpEMpdPeBsI/Mjx9uwtbFJMZBgYEQ5zL/5+Rq3hgXTZa5P8R8zTkUA/vRqP8OQie0wSQveiCbdh6pceURNz9NDo90u48XqDwnBDnU4qj8GmzbOZfER9SYcqJQQ==#|dyHWu6kTeS0x8C3ZqgBW2A==|nKeAAJk3eJT7aaq86SL9emoXdMIU7JRFn6B4n2rYg84aSWeylh4oTHbi41rjxWjQcZH0fj+ClsoKv+FWIYPcC1PjrGRv/GP12AoHT1BZKjMSHBG9VfJ+e+ND6OgYzW0wDk2ftfTUR5lu0EtfAiHcW3/8DdNoXXK60kWZgG+tkwcMEkJvbkUUfEKC/aTnsH3oRf+4L5+amBE7zPBql+yj7UdmbQeL3jCATmCZp83H+H4l2YZZkhY/fJhzk+2GL6Qsdj2s9F/BguALFlJxUb5jg9KXydCQ8OebglpkC/WJBWeWJhRIK6YowE1qXyugs/+hntMO75JhY07XdzIyu3qO5A==

Appendix C. FACEBOOK APPLICATION REGISTRATION

C.1 Creation Facebook Application “Privacy Solution Center”

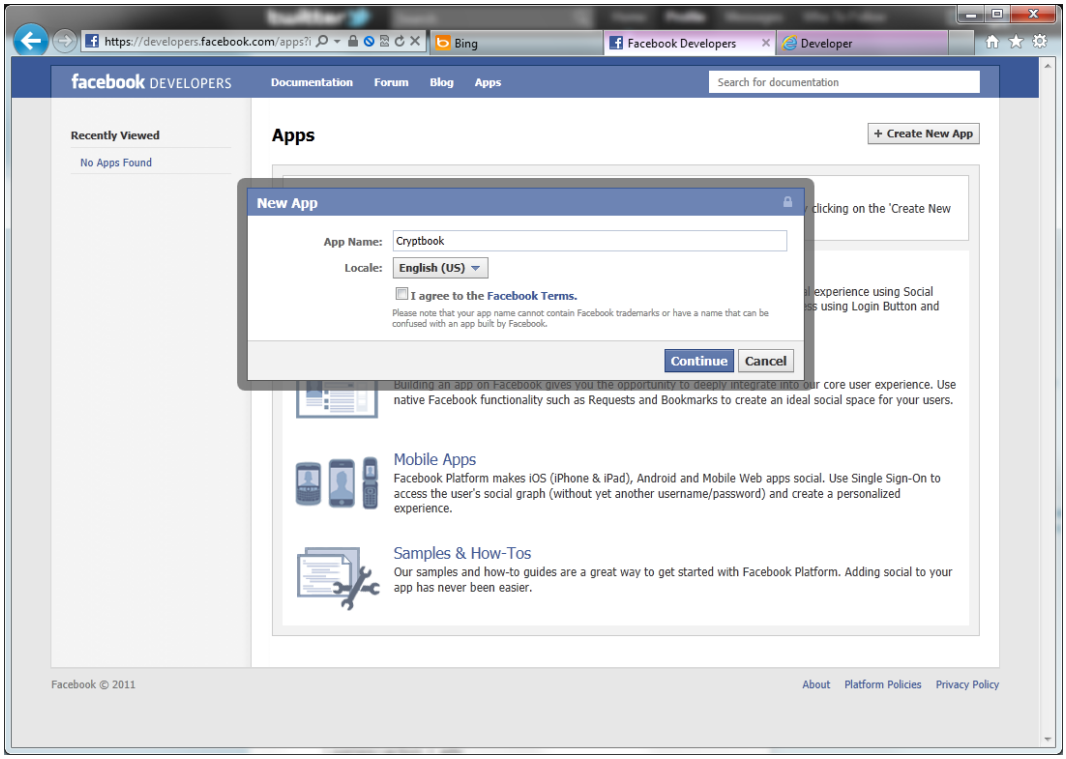


Figure 18 Facebook App Creation

Initially the name Cryptbook was planned to be used for the Facebook application, although this was not possible, since Facebook prevents the term book in an application name.

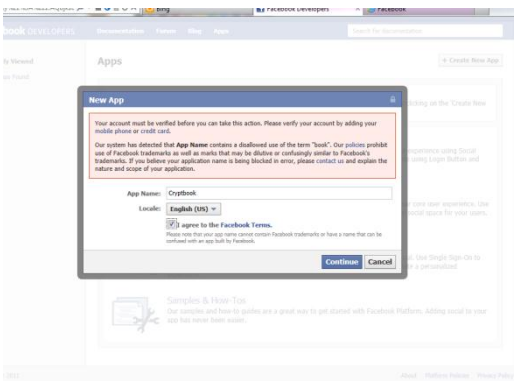


Figure 19 Facebook prevents creation of "Cryptbook"

C.2 Privacy Solution Center (Alpha) administrative page on Facebook

Below is a screenshot of the administrative view of the privacy solution center. For privacy reasons some fields are hidden intentionally.

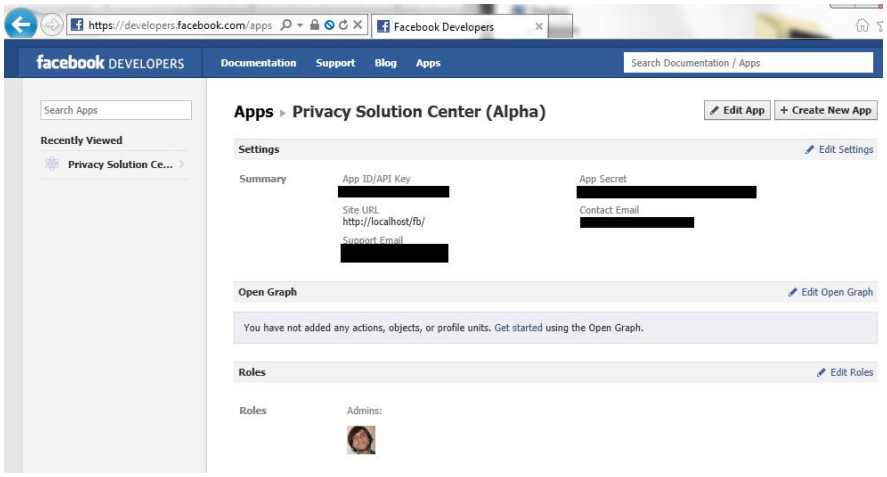


Figure 20 Privacy Solution Center App on Facebook Admin View

Appendix D. SOURCE CODE

D.1 Class Diagrams

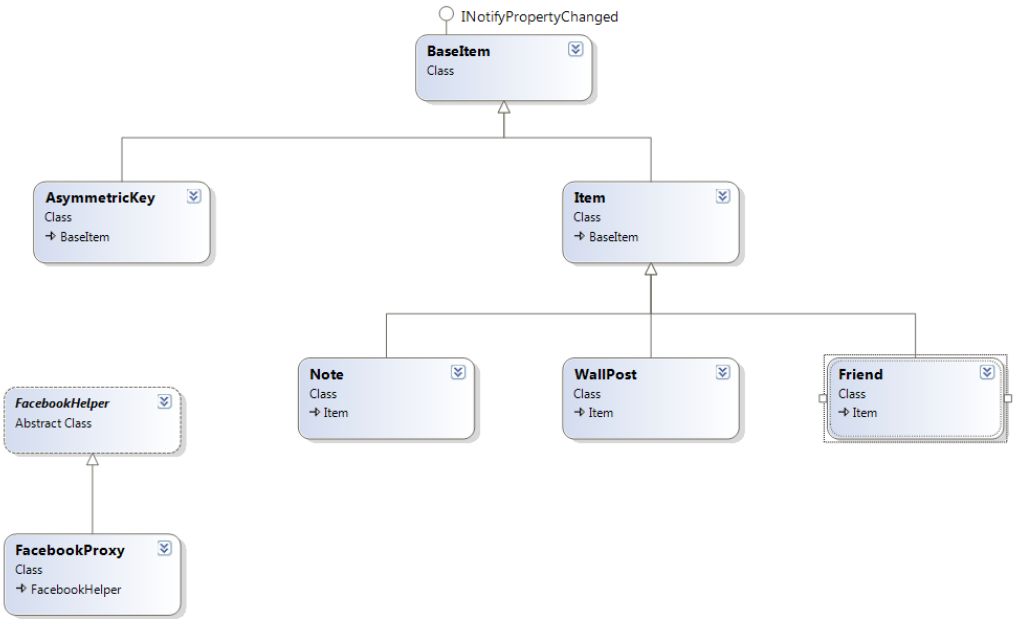


Figure 21 Data Contracts and FacebookHelper classes

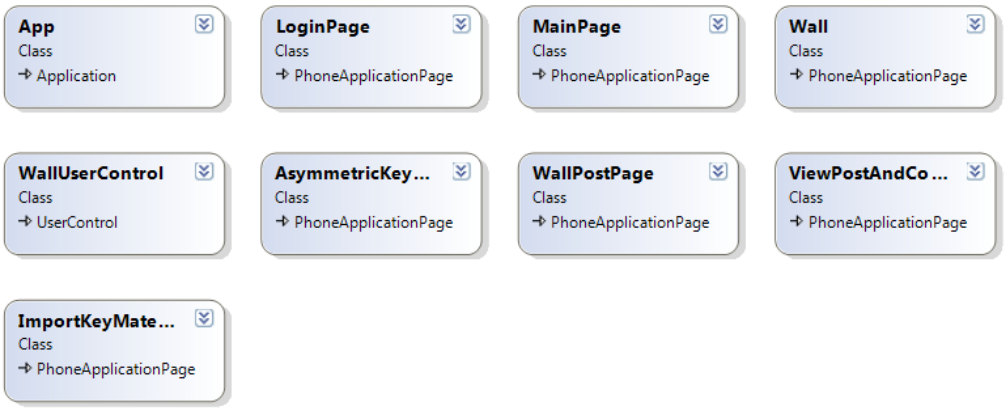
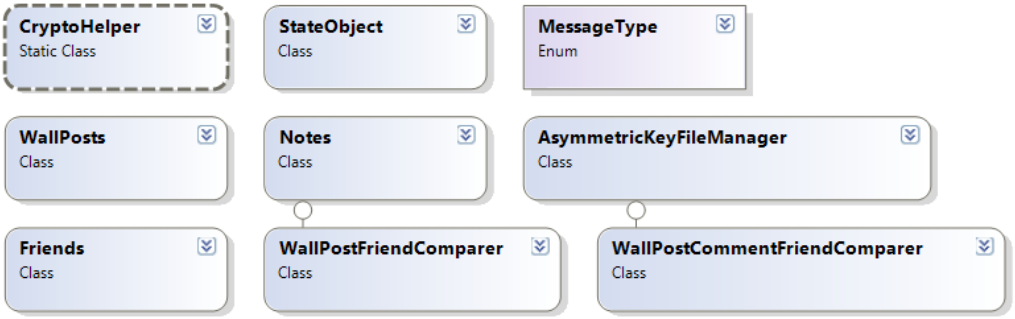


Figure 22 Helper classes



D.2 Sources code C# and XAML

The C# using statements are omitted for clarity purposes.

D.2.1 Cryptbook.DataContract.BaseItem

```
namespace Cryptbook.DataContracts
{
    [DataContract]
    public class BaseItem : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        protected void NotifyPropertyChanged(String propertyName)
        {
            PropertyChangedEventHandler handler = PropertyChanged;
            if (null != handler)
            {
                handler(this, new PropertyChangedEventArgs(propertyName));
            }
        }
    }
}
```

D.2.2 Cryptbook.DataContract.Item

```
namespace Cryptbook.DataContracts
{
    [DataContract]
    public class Item : BaseItem
    {
        private string id;

        [DataMember(Name = "id")]
        public string Id
        {
            get { return id ?? ""; }
            set { id = value; NotifyPropertyChanged("id"); }
        }

        /// <summary>
```

```
        /// custom data field not used by facebook
        /// </summary>
        private bool isEncrypted = false;
        public bool IsEncrypted
        {
            get { return isEncrypted; }
            set { isEncrypted = value; NotifyPropertyChanged("GetIsEncrypted");
}
        }

        /// <summary>
        /// Friendly return value for IsEncryptedProperty
        /// </summary>
        public string GetIsEncrypted
        {
            get
            {
                if (isEncrypted)
                {
                    return "encrypted";
                }
                else
                    return "plaintext";
            }
        }
    }
}
```

D.2.3 Cryptbook.DataContract.Friends

```
namespace Cryptbook.DataContracts
{
    //Based on the documentation at
http://developers.facebook.com/docs/reference/api/post/
    public class Friends
    {
        [DataContract]
        public class Feed
        {
            [DataMember(Name = "data")]
            public ObservableCollection<Friend> Friendlist = new ObservableCol-
lection<Friend>();
        }

         /// <summary>
        /// Manually serialize the object to JSON
        /// </summary>
        /// <param name="user"></param>
        /// <returns></returns>
        public static string SerializeJSON(Friends.Feed data)
        {
            //Create a stream to serialize the object to.
            MemoryStream memory = new MemoryStream();

            // Serializer the User object to the stream.
            DataContractJsonSerializer serializer = new DataContractJsonSerial-
izer(typeof(Friends.Feed));
            serializer.WriteObject(memory, data);
            byte[] jsonBytes = memory.ToArray();
            memory.Close();
```

```
            return Encoding.UTF8.GetString(jsonBytes, 0, jsonBytes.Length);
        }

        // Deserialize a JSON stream to a User object.
        public static Cryptbook.DataContracts.Friends.Feed Deserial-
izeJSON(string json)
        {
            Cryptbook.DataContracts.Friends.Feed deserializedData = new Crypt-
book.DataContracts.Friends.Feed();
            MemoryStream memory = new
MemoryStream(Encoding.UTF8.GetBytes(json));

            DataContractJsonSerializer deserializer = new DataContractJsonSeri-
alizer(deserializedData.GetType());
            deserializedData = deserializer.ReadObject(memory) as Crypt-
book.DataContracts.Friends.Feed;
            memory.Close();

            return deserializedData;
        }
    }

    [DataContract]
    public class Friend : Item
    {
        private string name;
        [DataMember(Name = "name")]
        public string Name
        {
            get { return name ?? ""; }
            set { name = value; NotifyPropertyChanged("Name"); }
        }

        private string picture;
        [DataMember(Name = "picture")]
        public string Picture
        {
            get { return picture ?? ""; }
            set { picture = value; NotifyPropertyChanged("Picture"); }
        }

        private string link;
        [DataMember(Name = "link")]
        public string Link
        {
            get { return link ?? ""; }
            set { link = value; NotifyPropertyChanged("Link"); }
        }


        public static string SerializeJSON(Friend data)
        {
            //Create a stream to serialize the object to.
            MemoryStream memory = new MemoryStream();

            // Serializer the User object to the stream.
            DataContractJsonSerializer serializer = new DataContractJsonSerial-
izer(typeof(Friend));
            serializer.WriteObject(memory, data);
            byte[] jsonBytes = memory.ToArray();
            memory.Close();


            return Encoding.UTF8.GetString(jsonBytes, 0, jsonBytes.Length);
        }
```

```
        // Deserialize a JSON stream to a User object.
        public static Cryptbook.DataContracts.Friend DeserializeJSON(string
json)
        {
            Cryptbook.DataContracts.Friend deserializedData = new Crypt-
book.DataContracts.Friend();
            MemoryStream memory = new
MemoryStream(Encoding.UTF8.GetBytes(json));

            DataContractJsonSerializer deserializer = new DataContractJsonSeri-
alizer(deserializedData.GetType());
            deserializedData = deserializer.ReadObject(memory) as Crypt-
book.DataContracts.Friend;
            memory.Close();

            return deserializedData;
        }
    }
}
```

D.2.4 Cryptbook.DataContract.Notes

```
namespace Cryptbook.DataContracts
{
    //Based on the documentation at
http://developers.facebook.com/docs/reference/api/post/
    public class Notes
    {
        [DataContract]
        public class Feed
        {
            [DataMember(Name = "data")]
            public ObservableCollection<Note> NoteList = new ObservableCollec-
tion<Note>();
        }

         /// <summary>
        /// Manually serialize the object to JSON
        /// </summary>
        /// <param name="user"></param>
        /// <returns></returns>
        public static string SerializeJSON(Notes.Feed data)
        {
            //Create a stream to serialize the object to.
            MemoryStream memory = new MemoryStream();

            // Serializer the User object to the stream.
            DataContractJsonSerializer serializer = new DataContractJsonSerial-
izer(typeof(Notes.Feed));
            serializer.WriteObject(memory, data);
            byte[] jsonBytes = memory.ToArray();
            memory.Close();


            return Encoding.UTF8.GetString(jsonBytes, 0, jsonBytes.Length);
        }

        // Deserialize a JSON stream to a User object.
        public static Cryptbook.DataContracts.Notes.Feed DeserializeJSON(string
json)
        {
```

```
            Cryptbook.DataContracts.Notes.Feed deserializedData = new Crypt-
book.DataContracts.Notes.Feed();
            MemoryStream memory = new
MemoryStream(Encoding.UTF8.GetBytes(json));

            DataContractJsonSerializer deserializer = new DataContractJsonSeri-
alizer(deserializedData.GetType());
            deserializedData = deserializer.ReadObject(memory) as Crypt-
book.DataContracts.Notes.Feed;
            memory.Close();

            return deserializedData;
        }
    }

    [DataContract]
    public class Note : Item
    {
        private string subject;
        [DataMember(Name = "subject")]
        public string Subject
        {
            get { return subject ?? ""; }
            set { subject = value; NotifyPropertyChanged("Subject"); }
        }

        private string message;
        [DataMember(Name = "message")]
        public string Message
        {
            get { return message ?? ""; }
            set { message = value; NotifyPropertyChanged("Message"); }
        }


        private Friend from = new Friend();
        [DataMember(Name = "from")]
        public Friend From
        {
            get { return from; }
            set { from = value; NotifyPropertyChanged("From"); }
        }

        public static string SerializeJSON(Note data)
        {
            //Create a stream to serialize the object to.
            MemoryStream memory = new MemoryStream();

            // Serializer the User object to the stream.
            DataContractJsonSerializer serializer = new DataContractJsonSerial-
izer(typeof(Note));
            serializer.WriteObject(memory, data);
            byte[] jsonBytes = memory.ToArray();
            memory.Close();


            return Encoding.UTF8.GetString(jsonBytes, 0, jsonBytes.Length);
        }

        // Deserialize a JSON stream to a User object.
        public static Cryptbook.DataContracts.Note DeserializeJSON(string json)
        {
            Cryptbook.DataContracts.Note deserializedData = new Crypt-
book.DataContracts.Note();
            MemoryStream memory = new
MemoryStream(Encoding.UTF8.GetBytes(json));
```

```
            DataContractJsonSerializer deserializer = new DataContractJsonSeri-
alizer(deserializedData.GetType());
            deserializedData = deserializer.ReadObject(memory) as Crypt-
book.DataContracts.Note;
            memory.Close();

            return deserializedData;
        }
    }
}
```

D.2.5 Cryptbook.DataContract.WallPosts

```
namespace Cryptbook.DataContracts
{
    public class WallPosts
    {
        [DataContract]
        public class Feed
        {
            [DataMember(Name = "data")]
            public ObservableCollection<WallPost> Wallposts = new Observable-
Collection<WallPost>();
        }

        /// <summary>
        /// Manually serialize the object to JSON
        /// </summary>
        /// <param name="user"></param>
        /// <returns></returns>
        public static string SerializeJSON(WallPosts.Feed data)
        {
            //Create a stream to serialize the object to.
            MemoryStream memory = new MemoryStream();

            // Serializer the User object to the stream.
            DataContractJsonSerializer serializer = new DataContractJsonSerial-
izer(typeof(WallPosts.Feed));
            serializer.WriteObject(memory, data);
            byte[] jsonBytes = memory.ToArray();
            memory.Close();


            return Encoding.UTF8.GetString(jsonBytes, 0, jsonBytes.Length);
        }

        // Deserialize a JSON stream to a User object.
        public static Cryptbook.DataContracts.WallPosts.Feed Deserial-
izeJSON(string json)
        {
            Cryptbook.DataContracts.WallPosts.Feed deserializedData = new
Cryptbook.DataContracts.WallPosts.Feed();
            MemoryStream memory = new
MemoryStream(Encoding.UTF8.GetBytes(json));

            DataContractJsonSerializer deserializer = new DataContractJsonSeri-
alizer(deserializedData.GetType());
            deserializedData = deserializer.ReadObject(memory) as Crypt-
book.DataContracts.WallPosts.Feed;
            memory.Close();
```

```
            return deserializedData;
        }
    }


    [DataContract]
    public class WallPost : Item
    {

        private Friend from = new Friend();
        [DataMember(Name = "from")]
        public Friend From
        {
            get { return from; }
            set { from = value; NotifyPropertyChanged("From"); }
        }


        private ToWhom to = new ToWhom();
        [DataMember(Name = "to")]
        public ToWhom To
        {
            get { return to; }
            set { to = value; NotifyPropertyChanged("To"); }
        }


        private string message;

        [DataMember(Name = "message")]
        public string Message
        {
            get { return message; }
            set { message = value; NotifyPropertyChanged("Message"); }
        }


        private string picture;
        [DataMember(Name = "picture")]
        public string Picture
        {
            get { return picture ?? ""; }
            set { picture = value; NotifyPropertyChanged("Picture"); }
        }


        private string link;
        [DataMember(Name = "link")]
        public string Link
        {
            get { return link ?? ""; }
            set { link = value; NotifyPropertyChanged("Link"); }
        }


        private string name;
        [DataMember(Name = "name")]
        public string Name
        {
            get { return name ?? ""; }
            set { name = value; NotifyPropertyChanged("Name"); }
        }


        private string caption;
        [DataMember(Name = "caption")]
```

```
public string Caption
{
    get { return caption ?? ""; }
    set { caption = value; NotifyPropertyChanged("Caption"); }
}


private string description;
[DataMember(Name = "description")]
public string Description
{
    get { return description ?? ""; }
    set { description = value; NotifyPropertyChanged("Description"); }
}


private string source;
[DataMember(Name = "source")]
public string Source
{
    get { return source ?? ""; }
    set { source = value; NotifyPropertyChanged("Source"); }
}


[DataMember(Name = "properties")]
public List<Property> properties = new List<Property>();



private string icon;
[DataMember(Name = "icon")]
public string Icon
{
    get { return icon ?? ""; }
    set { icon = value; NotifyPropertyChanged("Icon"); }
}


private List<Action> actions = new List<Action>();
[DataMember(Name = "actions")]
public List<Action> Actions
{
    get { return actions; }
    set { actions = value; NotifyPropertyChanged("Actions"); }
}


private Privacy privacy = new Privacy();

[DataMember(Name = "privacy")]
public Privacy PrivacyList
{
    get { return privacy; }
    set { privacy = value; NotifyPropertyChanged("PrivacyList"); }
}


private string type;
[DataMember(Name = "type")]
public string Type
{
    get { return type ?? ""; }
    set { type = value; NotifyPropertyChanged("Type"); }
}
```

```
        private Like likes;

        [DataMember(Name = "likes")]
        public Like Likes
        {
            get { return likes; }
            set { likes = value; NotifyPropertyChanged("Likes"); }
        }


        private Comments comments;
        [DataMember(Name = "comments")]
        public Comments CommentsList
        {
            get { return comments; }
            set { comments = value; NotifyPropertyChanged("CommentsList"); }
        }

        private int object_id;
        [DataMember(Name = "object_id")]
        public int Object_id
        {
            get { return object_id; }
            set { object_id = value; NotifyPropertyChanged("Object_id"); }

        }


        private ApplicationFacebook application = new ApplicationFacebook();
        [DataMember(Name = "application")]
        public ApplicationFacebook ApplicationName
        {
            get { return application; }
            set
            {
                application = value;
                NotifyPropertyChanged("Object_id");
            }
        }


        private string created_time;
        [DataMember(Name = "created_time")]
        public string CreatedTime
        {
            get { return created_time ?? ""; }
            set { created_time = value; NotifyPropertyChanged("CreatedTime"); }
        }

        public string GetFriendlyCreatedTime
        {
            get
            {
                return FacebookHelp-
er.GetFriendlyCreatedTimeHelper(created_time);
            }
        }


        private string updated_time;
        [DataMember(Name = "updated_time")]
        public string Updated_time
        {
            get { return updated_time ?? ""; }
```

```
            set { updated_time = value; NotifyPropertyChanged("Updated_time");
}
        }

        [DataMember(Name = "targeting")]
        private string targeting;
        public string Targeting
        {
            get { return targeting ?? ""; }
            set
            {
                targeting = value; NotifyPropertyChanged("Targeting");
            }
        }


        [DataContract]
        public class Privacy
        {
            private string value;

            [DataMember(Name = "value")]
            public string Value
            {
                get { return this.value ?? ""; }
                set { this.value = value; }
            }

            private string friends;

            [DataMember(Name = "friends")]
            public string Friends
            {
                get { return friends ?? ""; }
                set { friends = value; }
            }

            private string networks;

            [DataMember(Name = "networks")]
            public string Networks
            {
                get { return networks ?? ""; }
                set { networks = value; }
            }


            private string allow;

            [DataMember(Name = "allow")]
            public string Allow
            {
                get { return allow ?? ""; }
                set { allow = value; }
            }


            private string deny;

            [DataMember(Name = "deny")]
            public string Deny
            {
                get { return deny ?? ""; }
                set { deny = value; }
            }
```

```
    private string description;

    [DataMember(Name = "description")]
    public string Description
    {
        get { return description ?? ""; }
        set { description = value; }
    }
}

[DataContract]
public class ApplicationFacebook : Item
{

    private string name;

    [DataMember(Name = "name")]
    public string Name
    {
        get { return name; }
        set { name = value; NotifyPropertyChanged("Name"); }
    }
}

[DataContract]
public class Like
{
    private int count;
    [DataMember(Name = "count")]
    public int Count
    {
        get { return count; }
        set { count = value; }
    }

    private List<Friend> person;
    [DataMember(Name = "data")]
    public List<Friend> Person
    {
        get { return person; }
        set { person = value; }
    }



}

[DataContract]
public class Comments
{
    [DataMember(Name = "data")]
    public List<Comment> CommentsData { get; set; }

    public string GetCountForDisplay
    {
        get
        {
            if (CommentsData == null)
            {
                return string.Empty;
            }

            if (CommentsData.Count == 1)
            {
                return "1 comment";
            }
```

```
                if (CommentsData.Count > 1)
                {
                    return CommentsData.Count + " comments";
                }

                return string.Empty;
            }
        }

        public Visibility AreCommentsAvailable
        {
            get
            {
                if (CommentsData == null)
                    return Visibility.Collapsed;

                if (CommentsData.Count > 0)
                {
                    return Visibility.Visible;
                }
                else
                {
                    return Visibility.Collapsed;
                }
            }
        }
    }

    [DataContract]
    public class ToWhom
    {
        [DataMember(Name = "data")]
        public List<Friend> to;
    }

    [DataContract]
    public class Comment : Item
    {

        private Friend from;
        [DataMember(Name = "from")]
        public Friend From
        {
            get { return from; }
            set { from = value; NotifyPropertyChanged("From"); }
        }


        private string message;
        [DataMember(Name = "message")]
        public string Message
        {
            get { return message ?? ""; }
            set { message = value; NotifyPropertyChanged("Message"); }
        }


        private string created_time;
        [DataMember(Name = "created_time")]
        public string Created_time
        {
            get { return created_time ?? ""; }
            set { created_time = value; NotifyProperty-
Changed("Created_time"); }
        }
```

```
            public string GetFriendlyCreatedTime
            {
                get
                {
                    return FacebookHelp-
er.GetFriendlyCreatedTimeHelper(created_time);
                }
            }
        }

        [DataContract]
        public class Property : Item
        {
            private string name;
            [DataMember(Name = "name")]
            public string Name
            {
                get { return name ?? ""; }
                set { name = value; NotifyPropertyChanged("Name"); }
            }



            private string text;
            [DataMember(Name = "text")]
            public string Text
            {
                get { return text ?? ""; }
                set { text = value; NotifyPropertyChanged("Text"); }
            }
        }

        [DataContract]
        public class Action
        {

            private string name;

            [DataMember(Name = "name")]
            public string Name
            {
                get { return name; }
                set { name = value; }
            }


            private string link;

            [DataMember(Name = "link")]
            public string Link
            {
                get { return link; }
                set { link = value; }
            }
        }
    }
}
```

D.2.6 Cryptbook.App

```
namespace Cryptbook
{
    public partial class App : Application
    {
        public static bool IsUserAuthenticated = false;
        public static string AccessToken = string.Empty;

        public static AsymmetricKey PrivatePublicKey
        {
            get
            {
                return PrivatePublicKeyCollection.OrderByDescending(key =>
key.IssueDate).FirstOrDefault();
            }
        }

        public static ObservableCollection<AsymmetricKey> PrivatePublicKeyCol-
lection = new ObservableCollection<AsymmetricKey>();

        public static readonly string ApplicationName = "Project Cryptbook";


        public static DataContracts.Friend CurrentUser;
        public static DataContracts.Friends Friendlist;

        public static List<DataContracts.Friend> SelectedRecipients = new
List<Friend>();


        /// <summary>
        /// Provides easy access to the root frame of the Phone Application.
        /// </summary>
        /// <returns>The root frame of the Phone Application.</returns>
        public PhoneApplicationFrame RootFrame { get; private set; }

        /// <summary>
        /// Constructor for the Application object.
        /// </summary>
        public App()
        {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;

            // Standard Silverlight initialization
            InitializeComponent();

            // Phone-specific initialization
            InitializePhoneApplication();

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters
                Application.Current.Host.Settings.EnableFrameRateCounter =
true;

                // Show the areas of the app that are being redrawn in each
frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
```

```
                // which shows areas of a page that are handed off to GPU with
a colored overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization =
true;

                // Disable the application idle detection by setting the
UserIdleDetectionMode property of the
                // application's PhoneApplicationService object to Disabled.
                // Caution:- Use this under debug mode only. Application that
disables user idle detection will continue to run
                // and consume battery power when the user is not using the
phone.
                PhoneApplicationService.Current.UserIdleDetectionMode = IdleDe-
tectionMode.Disabled;
            }
        }

        // Code to execute when the application is launching (eg, from Start)
        // This code will not execute when the application is reactivated
        private void Application_Launching(object sender, LaunchingEventArgs e)
        {
        }

        // Code to execute when the application is activated (brought to fore-
ground)
        // This code will not execute when the application is first launched
        private void Application_Activated(object sender, ActivatedEventArgs e)
        {
            //TODO: consider loading data here (for refresh cases)
        }

        // Code to execute when the application is deactivated (sent to back-
ground)
        // This code will not execute when the application is closing
        private void Application_Deactivated(object sender, DeactivatedEven-
tArgs e)
        {
        }

        // Code to execute when the application is closing (eg, user hit Back)
        // This code will not execute when the application is deactivated
        private void Application_Closing(object sender, ClosingEventArgs e)
        {
            // Ensure that required application state is persisted here.
        }

        // Code to execute if a navigation fails
        private void RootFrame_NavigationFailed(object sender, Navigation-
FailedEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // A navigation has failed; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
        }

        // Code to execute on Unhandled Exceptions
        private void Application_UnhandledException(object sender, Applica-
tionUnhandledExceptionEventArgs e)
        {
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // An unhandled exception has occurred; break into the debugger
                System.Diagnostics.Debugger.Break();
            }
```

```
        }

        #region Phone application initialization

        // Avoid double-initialization
        private bool phoneApplicationInitialized = false;

        // Do not add any additional code to this method
        private void InitializePhoneApplication()
        {
            if (phoneApplicationInitialized)
                return;

            // Create the frame but don't set it as RootVisual yet; this allows
the splash
            // screen to remain active until the application is ready to ren-
der.
            RootFrame = new PhoneApplicationFrame();
            RootFrame.Navigated += CompleteInitializePhoneApplication;

            // Handle navigation failures
            RootFrame.NavigationFailed += RootFrame_NavigationFailed;

            // Ensure we don't initialize again
            phoneApplicationInitialized = true;
        }

        // Do not add any additional code to this method
        private void CompleteInitializePhoneApplication(object sender, Naviga-
tionEventArgs e)
        {
            // Set the root visual to allow the application to render
            if (RootVisual != RootFrame)
                RootVisual = RootFrame;

            // Remove this handler since it is no longer needed
            RootFrame.Navigated -= CompleteInitializePhoneApplication;
        }

        #endregion
    }
}
```

D.2.7 Cryptbook.AsymmetricKey

```
namespace Cryptbook
{

    public class AsymmetricKey : BaseItem
    {
        private string user;
        public string User { get {return user;} set { user = value; NotifyProp-
ertyChanged("User"); } }

        private string id;
        public string Id { get {return id;} set { id = value; NotifyProperty-
Changed("Id"); } }

        private string privatePublicKey;
```

```
        public string PrivatePublicKey { get { return privatePublicKey; } set {
privatePublicKey = value; NotifyPropertyChanged("PrivatePublicKey"); } }

        private string publicKey;
        public string PublicKey { get { return publicKey; } set { publicKey =
value; NotifyPropertyChanged("PublicKey"); } }

        private string issueDate;
        public string IssueDate { get { return issueDate; } set { issueDate =
value; NotifyPropertyChanged("IssueDate"); } }

        public static string Backup(AsymmetricKey key)
        {
            //    XDocument xml = null;

            //    if (includePrivateKey)
            //    {
            //        xml = new XDocument(
            //            new XElement("cryptbook",
            //                new XComment("key material - JR"),
            //                new XElement("version", "1"),
            //                new XElement("guid", Id),
            //                new XElement("privatepublickey", PrivatePublicK-
ey),
            //                new XElement("public", PublicKey)
            //            )
            //        );


            //    }
            //    else
            //    {
            //        xml = new XDocument(
            //            new XElement("cryptbook",
            //                new XComment("key material - JR"),
            //                new XElement("version", "1"),
            //                new XElement("guid", Id),
            //                new XElement("privatepublickey", ""),
            //                new XElement("public", PublicKey)
            //            )
            //        );
            //    }

            //    return xml.ToString();



            MemoryStream memory = new MemoryStream();

            XmlSerializer serializer = new XmlSerializ-
er(typeof(AsymmetricKey));
            serializer.Serialize(memory, key);

            byte[] content = memory.ToArray();
            string xml = Encoding.UTF8.GetString(content, 0, content.Length);
            return xml;
        }

        public static AsymmetricKey Restore(string xml)
        {

            AsymmetricKey key = new AsymmetricKey();

            MemoryStream memory = new
MemoryStream(Encoding.UTF8.GetBytes(xml));
```

```
            XmlSerializer serializer = new XmlSerializ-
er(typeof(AsymmetricKey));
            key = (AsymmetricKey)serializer.Deserialize(memory);

            return key;
        }
    }



    public class AsymmetricKeyFileManager
    {
        private static string Filename
        {
            get
            {
                return prefix + ".asymkey.xml";
            }
        }

        private static string prefix = String.Empty;
        public static ObservableCollection<AsymmetricKey> Load(string filepre-
fix)
        {
            prefix = fileprefix;

            ObservableCollection<AsymmetricKey> keys = null;

            using (IsolatedStorageFile isf = IsolatedStor-
ageFile.GetUserStoreForApplication())
            {
                if (isf.FileExists(Filename))
                {
                    using (IsolatedStorageFileStream isfs = new IsolatedStor-
ageFileStream(Filename, FileMode.Open, isf))
                    {
                        using (StreamReader reader = new StreamReader(isfs))
                        {
                            if (!reader.EndOfStream)
                            {
                                XmlSerializer serializer = new XmlSerializ-
er(typeof(ObservableCollection<AsymmetricKey>));
                                keys = (ObservableCollec-
tion<AsymmetricKey>)serializer.Deserialize(reader);
                            }
                        }
                    }
                }
            }

            if (keys == null)
            {
                keys = new ObservableCollection<AsymmetricKey>();
            }

            return keys;
        }

        public static void Save(ObservableCollection<AsymmetricKey> keys,
string fileprefix)
        {
            using (IsolatedStorageFile isf = IsolatedStor-
ageFile.GetUserStoreForApplication())
            {
                using (IsolatedStorageFileStream isfs = new IsolatedStor-
ageFileStream(Filename, FileMode.Create, isf))
```

```
                {
                    using (StreamWriter writer = new StreamWriter(isfs))
                    {
                        XmlSerializer serializer = new XmlSerializ-
er(typeof(ObservableCollection<AsymmetricKey>));
                        serializer.Serialize(writer, keys);
                        writer.Flush();
                        writer.Close();
                    }
                }
            }
        }

    }
}
```

D.2.8 Cryptbook.AsymmetricKeyManager

```
namespace Cryptbook
{
    public partial class AsymmetricKeyManager : PhoneApplicationPage
    {
        private string facebookId;
        private string facebookName;

        public AsymmetricKeyManager()
        {
            InitializeComponent();

            UpdateView();
        }


        private void UpdateView()
        {
            AsymmetricKeyListBox.DataContext = App.PrivatePublicKeyCollection;

            if (App.PrivatePublicKey == null)
                return;

            if (App.PrivatePublicKey.User == App.CurrentUser.Id)
            {
                KeyGeneratedStackPanel.Visibility = Sys-
tem.Windows.Visibility.Visible;
                PublicKey.Text = App.PrivatePublicKey.PublicKey;

                ActiveKeyIssueDate.Text = "Issued: " + Con-
vert.ToDateTime(App.PrivatePublicKey.IssueDate).ToString("r");
                ActiveKey.Text = App.PrivatePublicKey.Id;
            }
            else
            {
                KeyGeneratedStackPanel.Visibility = Sys-
tem.Windows.Visibility.Collapsed;
            }
        }
```

```
        protected override void OnNavi-
gatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            facebookId = NavigationContext.QueryString.Where(item => item.Key
== "id").FirstOrDefault().Value;
            facebookName = NavigationContext.QueryString.Where(item => item.Key
== "name").FirstOrDefault().Value;

            base.OnNavigatedTo(e);
        }

        private void GenerateKeyPair_TextInput(object sender, TextComposition-
EventArgs e)
        {

        }

        private void GenerateKeyPair_Click(object sender, RoutedEventArgs e)
        {
            //if (App.PrivatePublicKey.User == App.CurrentUser.Id)
            //{
            //    if(MessageBox.Show("Do you want to create a new key pair and
overwrite the exising one? Make sure to backup the old key pair otherwise you
will not be able to decrypt information!",
            //        "Warning!", MessageBoxButton.OKCancel) == MessageBoxRe-
sult.Cancel)
            //    {
            //      return;
            //    }
            //}

            RSACryptoServiceProvider rsa = new RSACryptoServiceProvider(2048);
            AsymmetricKey key = new AsymmetricKey();

            key.Id = Guid.NewGuid().ToString();
            key.User = App.CurrentUser.Id;
            key.IssueDate = DateTime.UtcNow.ToString("s");

            //another option, let's url encode it - maybe that works with face-
book
            //exportcspparameters throws an exception so using xml here
            //facebook doesn't handle the xml tags well... so they will be ur-
lencoded before posting
            key.PrivatePublicKey = rsa.ToXmlString(true);
            key.PublicKey = rsa.ToXmlString(false);

            PublicKey.Text = key.PublicKey;

            App.PrivatePublicKeyCollection.Add(key);
            AsymmetricKeyFileManager.Save(App.PrivatePublicKeyCollection,
App.CurrentUser.Id);

            App.PrivatePublicKeyCollection = AsymmetricKeyFileManag-
er.Load(App.CurrentUser.Id);

            UpdateView();

        }



        private void PostPublicKey_Click(object sender, RoutedEventArgs e)
        {

            FacebookProxy proxy1 = new FacebookProxy( App.AccessToken);
```

```
            ///define storag format
            ///@CB|Version|KeyGUID|Base64{Key}|FacebookUserID
            ///

            //note: xml of public and private key are urlencoded - facebook
doesn't handle the xml tags well and removes them when reading the key back in

            string postMessagePublic= string.Format("@CB|1|{0}|{1}|{2}",
App.PrivatePublicKey.Id, HttpUtility.UrlEncode(App.PrivatePublicKey.PublicKey),
App.PrivatePublicKey.User);
            proxy1.PostNote(facebookId, FacebookHelp-
er.PublicKeyIdentifierOnFacebookNotes + " - Public Key - (do not alter)",
postMessagePublic, PostNote_UploadStringCompleted);

            //FacebookProxy proxy2 = new FacebookProxy(App.AccessToken);
            //string postMessagePrivate = string.Format("@CB|1|{0}|{1}|{2}",
App.PrivatePublicKey.Id, HttpUtili-
ty.UrlEncode(App.PrivatePublicKey.PrivatePublicKey),
App.PrivatePublicKey.User);
            //proxy2.PostNote(facebookId, FacebookHelp-
er.PublicKeyIdentifierOnFacebookNotes + " - Private debug - (do not alter)",
postMessagePrivate, PostNote_UploadStringCompleted);

            NavigationService.GoBack();
        }



        private void Post-
Note_UploadStringCompleted(UploadStringCompletedEventArgs e)
        {
            lock (e)
            {
                if (e.Error != null)
                {
                    MessageBox.Show("Error occured: " + e.Error.Message);
                }
                else
                {
                    string objectID = e.Result;
                    MessageBox.Show("Key succesfully published to your Facebook
Notes");
                }
            }
        }

        private void ImportPrivatePublicKey_Click(object sender, RoutedEven-
tArgs e)
        {
            NavigationService.Navigate(new Uri("/ImportKeyMaterial.xaml",
UriKind.Relative));
        }

        private void ExportKeyMaterial_Click(object sender, RoutedEventArgs e)
        {
            Microsoft.Phone.Tasks.EmailComposeTask email = new Mi-
crosoft.Phone.Tasks.EmailComposeTask();
            email.Subject = "key material - " + App.PrivatePublicKey.Id;
            email.Body = AsymmetricKey.Backup(App.PrivatePublicKey);
            email.Show();
        }

        private void CopyMaterial_Click(object sender, RoutedEventArgs e)
        {
               Clipboard.SetText(AsymmetricKey.Backup(App.PrivatePublicKey));
```

```
        }

        private void PostPrivateKey_Click(object sender, RoutedEventArgs e)
        {
            FacebookProxy proxy2 = new FacebookProxy(App.AccessToken);
            string postMessagePrivate = string.Format("@CB|1|{0}|{1}|{2}",
App.PrivatePublicKey.Id, HttpUtili-
ty.UrlEncode(App.PrivatePublicKey.PrivatePublicKey),
App.PrivatePublicKey.User);
            proxy2.PostNote(facebookId, FacebookHelp-
er.PublicKeyIdentifierOnFacebookNotes + " - Private debug - (do not alter)",
postMessagePrivate, PostNote_UploadStringCompleted);

        }
    }
}
```

D.2.9 Cryptbook.CryptoHelper

```
namespace Cryptbook
{
    public static class CryptoHelper
    {
        /// <summary>
        /// Encrypts the given plain text using a passphrase and given initial-
ization vector.
        /// The key generation is according to RFC 2898.
        /// </summary>
        /// <param name="passphrase">The passphrase to create the symmetric en-
cryption key from and subsequently encrypt the message</param>
        /// <param name="data">The plain text to encrypt</param>
        /// <param name="initializationVector">The initialization vector for
the AES encryption</param>
        /// <returns></returns>
        public static string Encrypt(string passphrase, string data, string in-
itializationVector)
        {
            Rfc2898DeriveBytes key = new Rfc2898DeriveBytes(passphrase,
                Encoding.Unicode.GetBytes(initializationVector),
                1000);

            AesManaged aes = new AesManaged();
            aes.Key = key.GetBytes(aes.KeySize / 8);
            aes.IV = Convert.FromBase64String(initializationVector);

            byte[] cipherTextBuffer;
            using (MemoryStream memoryStream = new MemoryStream())
            {
                using (CryptoStream cryptoStream = new Cryp-
toStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write))
                {
                    byte[] clearTextBuffer = Encoding.Unicode.GetBytes(data);
                    cryptoStream.Write(clearTextBuffer, 0, clearTextBuff-
er.Length);
                    cryptoStream.FlushFinalBlock();
                }

                cipherTextBuffer = memoryStream.ToArray();
            }
```

```
            return Convert.ToBase64String(cipherTextBuffer);
        }

        /// <summary>
        /// Decrypts the given cipher text using a passphrase and given ini-
tialization vector.
        /// The key generation is according to RFC 2898.
        /// </summary>
        /// <param name="passphrase">The passphrase to create the symmetric en-
cryption key from and subsequently encrypt the message</param>
        /// <param name="cipherText">The base64 encoded cipher text to de-
crypt</param>
        /// <param name="initializationVector">The initialization vector for
the AES encryption</param>
        /// <returns></returns>
        public static string Decrypt(string passphrase, string cipherText,
string initializationVector)
        {
            Rfc2898DeriveBytes key = new Rfc2898DeriveBytes(passphrase,
                                                            Encod-
ing.Unicode.GetBytes(initializationVector),
                                                            1000);

            AesManaged aes = new AesManaged();
            aes.Key = key.GetBytes(aes.KeySize / 8);
            aes.IV = Convert.FromBase64String(initializationVector);

            byte[] cipherTextBuffer = Convert.FromBase64String(cipherText);
            byte[] clearTextBuffer;

            using (MemoryStream memory = new MemoryStream())
            {
                using (CryptoStream cryptoStream = new CryptoStream(memory,
aes.CreateDecryptor(), CryptoStreamMode.Write))
                {
                    cryptoStream.Write(cipherTextBuffer, 0, cipherTextBuff-
er.Length);
                    cryptoStream.FlushFinalBlock();
                }

                clearTextBuffer = memory.ToArray();
            }

            return Encoding.Unicode.GetString(clearTextBuffer, 0, clearText-
Buffer.Length);
        }


       public static string EncryptAsymmetric(List<AsymmetricKey> publicKeyMa-
terial, AsymmetricKey privateKeyMaterial, string message)
       {
            RSACryptoServiceProvider privateKey = new RSACryptoServiceProvid-
er();
            privateKey.FromXmlString(privateKeyMaterial.PrivatePublicKey);

            AesManaged aes = new AesManaged();
            aes.GenerateKey();
            aes.GenerateIV();
            byte[] aesCipherTextBufferMessage;

            //use the symmetric key to encrypt the message
            using (MemoryStream memoryStream = new MemoryStream())
            {
                using (CryptoStream cryptoStream = new Cryp-
toStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write))
                {
```

```
                byte[] clearTextBuffer = Encoding.Unicode.GetBytes(message);
                cryptoStream.Write(clearTextBuffer, 0, clearTextBuff-
er.Length);
                cryptoStream.FlushFinalBlock();
            }

            aesCipherTextBufferMessage = memoryStream.ToArray();
        }

        //Format of the final message
        //#CB| Version | aes(message) |
#rsakeyguid:rsa1(aeskey)#rsakeyguid:rsa2(aeskey)...# | aesiv | signature

        string recipients = String.Empty;
        foreach(AsymmetricKey key in publicKeyMaterial)
        {
            RSACryptoServiceProvider publicKey = new RSACryptoServiceProvid-
er();
            byte[] rsaEncryptedSymmetricKeyCipher = null;

            publicKey.FromXmlString(key.PublicKey);
            rsaEncryptedSymmetricKeyCipher = publicKey.Encrypt(aes.Key,
false);

            recipients += key.Id + ":" + Con-
vert.ToBase64String(rsaEncryptedSymmetricKeyCipher) + "#";
        }

        //construct the message
        string envelopedMessage = string.Format("#CB|{0}|{1}|{2}|{3}",
            "1",
            Convert.ToBase64String(aesCipherTextBufferMessage),
            recipients,   //note can contain multiple encryption keys seper-
ate the #
            Convert.ToBase64String(aes.IV));  //don't need to encrypt the
IV???

        //finally sign and append the signature to the message
        byte[] hash = private-
Key.SignData(Encoding.Unicode.GetBytes(envelopedMessage), "SHA256");
        envelopedMessage = envelopedMessage + "|" + Con-
vert.ToBase64String(hash);

        return envelopedMessage;
    }

    public static string DecryptAsymmetric(List<AsymmetricKey> privateKey-
ToDecrypt, AsymmetricKey publicKeyOfSigner, string message)
    {
        RSACryptoServiceProvider publicKeySigner = new RSACryptoServicePro-
vider();
        publicKeySigner.FromXmlString(publicKeyOfSigner.PublicKey);

        //#CB| Version | aes(message) |
rsakeyguid:rsa1(aeskey)#rsakeyguid:rsa2(aeskey)...@ | aesiv | digest
        string[] parts = message.Split('|');
        if ( parts.Length !=6)
            throw new InvalidProgramException("invalid message structure");


        //validate the signature
        string signedMessage = message.Substring(0, mes-
sage.LastIndexOf('|'));
        byte[] clearTextBytes = Encoding.Unicode.GetBytes(signedMessage);

        //now decrypt the signature
```

```
            //what is OAEP padding????
            if (publicKeySigner.VerifyData(clearTextBytes, "SHA256", Con-
vert.FromBase64String(parts[5])) == false)
            {
                return "Message signature is invalid. The message is not from "
+ publicKeyOfSigner.User; //throw new CryptographicException("Message signature
is invalid. The message is not from " + publicKeyOfSigner.User);
            }

            //check if the message was encrypted with a public key that matches
the private one
            RSACryptoServiceProvider privateKeyDecryptor;
            AesManaged aes = null;

            bool foundKey = false;
            string[] recipients = parts[3].Split('#');

            //enumerate through all the recipient keys
            foreach (string recipient in recipients)
            {
                string[] tuple = recipient.Split(':');

                // cross check each recipient with all the id's of the this user
has
                privateKeyToDecrypt.ForEach(privateKey =>
                    {
                        if(foundKey)
                            return;

                        //if the guid matches we have a winner and we should be
able to decrypt
                        //the message, so load up the RSACryptoProvider and de-
crypt the AES key
                        if(privateKey.Id == tuple[0])
                        {
                            privateKeyDecryptor = new RSACryptoServiceProvid-
er();
                            privateKeyDecryp-
tor.FromXmlString(privateKey.PrivatePublicKey);
                            aes = new AesManaged();
                            aes.Key = privateKeyDecryp-
tor.Decrypt(Convert.FromBase64String(tuple[1]), false);
                            aes.IV = Convert.FromBase64String(parts[4]);

                            foundKey = true;
                        }
                    });

                //no need to check further, as soon as we found a key
                if (foundKey == true)
                    break;
            }

            if (!foundKey)
            {
                return "No valid decryption key was found for this message.";
            }

            if (aes == null)
                return "No valid decryption key was found for this message.";

            //now decrypt the message with the symmetric AES key
            byte[] cipherTextBuffer = Convert.FromBase64String(parts[2]);
            byte[] clearTextBuffer;

            using (MemoryStream memory = new MemoryStream())
```

```
            {
                using (CryptoStream cryptoStream = new CryptoStream(memory,
aes.CreateDecryptor(), CryptoStreamMode.Write))
                {
                    cryptoStream.Write(cipherTextBuffer, 0, cipherTextBuff-
er.Length);
                    cryptoStream.FlushFinalBlock();
                }

                clearTextBuffer = memory.ToArray();
            }

            string clearText = Encoding.Unicode.GetString(clearTextBuffer, 0,
clearTextBuffer.Length);
            return clearText;
        }

         public static string GetRandomNumberAsBase64()
         {
             byte[] number = new byte[16];

             RNGCryptoServiceProvider generator = new RNGCryptoServiceProvid-
er();
             generator.GetBytes(number);
             return Convert.ToBase64String(number);
         }

    }
}
```

D.2.10 Cryptbook.FacebookHelper

```
namespace Cryptbook
{

    public abstract class FacebookHelper
    {

        /// <summary>
        /// Endpoint to initiate client side OAuth2 flow with Facebook
        /// </summary>
        private static string facebookOAuth2Endpoint =
@"https://m.facebook.com/dialog/oauth?client_id={0}&redirect_uri={1}&scop
e={2}&response_type=token";
        public static string AppID = "<removed for security reasons>";
        private static string clientSecret = "<removed for security rea-
sons>";
        private static string scope =
"read_stream,publish_stream,user_notes,friends_notes";
        public static readonly string PublicKeyIdentifierOnFacebookNotes
= @"@Privacy Solution Center";

        public static string RedirectURL =
"https://www.facebook.com/connect/login_success.html";

        public static string GetOAuth2Endpoint
        {
            get
            {
                return
string.Format(FacebookHelper.facebookOAuth2Endpoint,
                    FacebookHelper.AppID,
                    FacebookHelper.RedirectURL,
```

```
                    FacebookHelper.scope);
            }
        }


        public static string GetLogoutEndpoint(string auth_token)
        {
            return
string.Format("https://www.facebook.com/logout.php?next={0}&access_token=
{1}", "https://localhost/fb", auth_token);
        }


        public static AsymmetricKey ExtractPublicKeyFromJsonRe-
sponse(string jsonResult)
        {
            ///continue deserialze object
            DataContracts.Notes.Feed notes = DataCon-
tracts.Notes.DeserializeJSON(jsonResult);

            AsymmetricKey key = new AsymmetricKey();
            string publicKeyIdentifier = FacebookHelp-
er.PublicKeyIdentifierOnFacebookNotes + " - Public";

            if (notes.NoteList.Count < 1)
            {
                throw new Exception("No public key found.");
            }

            DataContracts.Note note = null;
            foreach (Note n in notes.NoteList)
            {
                if (n.Subject.StartsWith(publicKeyIdentifier))
                {
                    note = n;
                    break;
                }
            }

            if (note == null)
            {
                throw new Exception("No public key found.");
            }

            //couldn't use extension metho where here - don't know why?
            //DataContracts.Note note = notes.NoteList.Where(n =>
n.Subject.StartsWith(publicKeyIdentifier)).FirstOrDefault();

            ///@CB|Version|KeyGUID|UrlEncoded{Key}|FacebookUserID
            string[] parts = note.Message.Split('|');

            //Facebook wraps the content in a <p>, we remove this if it's
present
            if (parts[0].StartsWith("<p>") == true)
            {
                parts[0] = parts[0].Replace("<p>", "");
            }

            if (parts[4].EndsWith("</p>") == true)
            {
                parts[4] = parts[4].Replace("</p>", "");
            }

            if (parts.Length != 5)
                throw new InvalidProgramException("not a valid key");
```

```
            if (note.From.Id != parts[4])
                throw new InvalidProgramException("User that posted the
public key is different from public key metadata");

            key.PublicKey = HttpUtility.UrlDecode(parts[3]);
            key.Id = parts[2];
            key.User = parts[4];

            return key;
        }


        private WebClient request = new WebClient();

        public FacebookHelper()
        {
            request.UploadStringCompleted += new UploadStringCompletedEv-
entHandler(request_UploadStringCompleted);
            request.DownloadStringCompleted += new DownloadStringCom-
pletedEventHandler(request_DownloadStringCompleted);
        }

        public void Post(Uri uri, string body, Ac-
tion<UploadStringCompletedEventArgs> callback)
        {

            StateObject state = new StateObject();
            state.callback = callback;

            request.UploadStringAsync(uri, "POST", body, state);
        }

        public void Get(Uri uri, Action<DownloadStringCompletedEventArgs>
callback, StateObject state)
        {
            if (state == null)
            {
              state = new StateObject();
            }

            if (state.callback == null)
            {
                state.callback = callback;
            }

            request.DownloadStringAsync(uri, state);
        }



        private void request_UploadStringCompleted(object sender, Up-
loadStringCompletedEventArgs r)
        {
            var callback = (Action<UploadStringCompletedEventArgs>)
((StateObject)r.UserState).callback;
            callback(r);
        }

        private void request_DownloadStringCompleted(object sender, Down-
loadStringCompletedEventArgs r)
        {
            var callback = (Ac-
tion<DownloadStringCompletedEventArgs>)((StateObject)r.UserState).callbac
k;
            callback(r);
        }
```

```
        public static string GetFriendlyCreatedTimeHelper(string creat-
ed_time)
        {
            if (created_time == null)
            {
                return string.Empty;
            }
            DateTime posttime = DateTime.Parse(created_time);  //this
seems to convert to local time
            //TimeSpan diff = DateTime.UtcNow.Subtract(posttime);
            TimeSpan diff = DateTime.Now.Subtract(posttime);

            if (diff.TotalSeconds < 60)
            {
                return "Few seconds ago.";
            }
            else if (diff.TotalMinutes < 2)
            {
                return "One minute ago.";
            }
            else if (diff.TotalMinutes < 60)
            {
                return (int)diff.TotalMinutes + " minutes ago.";
            }
            else if (diff.TotalHours < 2)
            {
                return "One hour ago.";
            }
            else if (diff.TotalHours < 24)
            {
                return (int)diff.TotalHours + " hours ago.";
            }
            else if (diff.TotalDays < 2)
            {
                return "A day ago.";
            }
            else if (diff.TotalDays < 30)
            {
                return (int)diff.TotalDays + " days ago.";
            }
            else if (diff.TotalDays < 365)
            {
                int months = (int)diff.TotalDays / 30;
                return (int)months + " months ago.";
            }
            else if (diff.TotalDays > 365)
            {
                return (int)diff.TotalDays + " days ago.";
            }
            else
            {
                return "Posted in " + posttime.Year;
            }
        }
    }


    public  enum MessageType
        {
            Wallpost,
            Comment
        }

    public class StateObject
```

```
        {
            public object callback;
            public WallPost.Comment Comment;
            public WallPost WallPost;
            public List<AsymmetricKey> PrivateKeys;
            public MessageType Type;
            public DataContracts.Friend FacebookUser;
        }
    }
```

D.2.11 Cryptbook.FacebookProxy

```
namespace Cryptbook
{
    public class FacebookProxy : FacebookHelper
    {
        private string accessToken;

        public FacebookProxy(string accessToken)
        {
            this.accessToken = accessToken;
        }

        public void GetUserInfo(string facebookId, Ac-
tion<DownloadStringCompletedEventArgs> callback)
        {
            string resource =
string.Format("https://graph.facebook.com/{0}?fields=id,name,picture,link,email
&access_token={1}", facebookId, accessToken);
            Uri uri = new Uri(resource);

            Get(uri, callback, null);
        }


        public void GetWallAsync(string facebookId, DateTime postsUntil, Ac-
tion<DownloadStringCompletedEventArgs> callback)
        {
            int resultLimit = 20;

            string until = postsUntil.ToUniversalTime().ToString("s");
            string request =
string.Format("https://graph.facebook.com/{0}/feed?access_token={1}&limit={2}&u
ntil={3}&fields=id,message,name,description,application,from,comments",
                    facebookId,
                    accessToken,
                    resultLimit,
                    until);

            Uri uri = new Uri(request);
            Get(uri, callback, null);
        }


        //public void GetWallAsync(string facebookId, int resultOffset,
DateTime since, Action<DownloadStringCompletedEventArgs> callback)
        //{
        //    int resultLimit = 20;
```

```
        //    string request =
string.Format("https://graph.facebook.com/{0}/feed?access_token={1}&limit={2}&o
ffset={3}&since={4}&fields=id,message,name,description,application,from,comment
s",
        //            facebookId,
        //            accessToken,
        //            resultLimit,
        //            resultOffset,
        //            since.ToUniversalTime());

        //    Uri uri = new Uri(request);
        //    Get(uri, callback, null);
        //}

        public void GetPictureOfUser(string facebookId, Ac-
tion<DownloadStringCompletedEventArgs> callback)
        {
            string resource =
string.Format("https://graph.facebook.com/{0}?access_token={1}&fields=id,name,p
icture",
                              facebookId,
                              accessToken);

            Uri uri = new Uri(resource);
            Get(uri, callback, null);
        }

        public void GetMyFriendsAsync(Action<DownloadStringCompletedEventArgs>
callback)
        {
            Uri uri = new
Uri("https://graph.facebook.com/me/friends?fields=id,name,picture,link,email&ac
cess_token=" + accessToken);
            Get(uri, callback, null);
        }

        public void GetPublicKey(string facebookId, Ac-
tion<DownloadStringCompletedEventArgs> callback, StateObject userdata)
        {
            string resource =
string.Format("https://graph.facebook.com/{0}/notes?access_token={1}&fields=id,
subject,message,from",
                facebookId,
                accessToken);

            Uri uri = new Uri(resource);

            Get(uri, callback, userdata);
        }


        public void PostNote(string facebookId, string subject, string message,
Action<UploadStringCompletedEventArgs> callback)
        {
            string wallPost =
string.Format("https://graph.facebook.com/{0}/notes?access_token={1}",
                facebookId,
                accessToken);

            Uri uri = new Uri(wallPost);

            string postBody = string.Format("subject={0}&message={1}",
HttpUtility.UrlEncode(subject), HttpUtility.UrlEncode(message));

            Post(uri, postBody, callback);
        }
```

```
        public void PostWall(string facebookId, string message, Ac-
tion<UploadStringCompletedEventArgs> callback)
        {
            string wallPost =
string.Format("https://graph.facebook.com/{0}/feed?access_token={1}",
                facebookId,
                accessToken);

            Uri uri = new Uri(wallPost);

            string postBody = string.Format("message={0}&application={1}",
HttpUtility.UrlEncode(message), App.ApplicationName);

            Post(uri, postBody, callback);
        }


        public void PostComment(string objectId, string message, Ac-
tion<UploadStringCompletedEventArgs> callback)
        {
            string wallPost =
string.Format("https://graph.facebook.com/{0}/comments?access_token={1}",
                objectId,
                accessToken);

            Uri uri = new Uri(wallPost);

            string postBody = string.Format("message={0}&application={1}",
HttpUtility.UrlEncode(message), App.ApplicationName);

            Post(uri, postBody, callback);
        }
    }
}
```

D.2.12 Cryptbook.ImportKeyMaterial

```
namespace Cryptbook
{
    public partial class ImportKeyMaterial : PhoneApplicationPage
    {
        public ImportKeyMaterial()
        {
            InitializeComponent();
        }

        private void ImportButton_Click(object sender, RoutedEventArgs e)
        {
            AsymmetricKey importKey = AsymmetricK-
ey.Restore(KeyTextBox.Text);
            if (App.PrivatePublicKeyCollection.Where(key => key.Id == im-
portKey.Id).Count() == 0)
            {
                App.PrivatePublicKeyCollection.Add(importKey);
                AsymmetricKeyFileManag-
er.Save(App.PrivatePublicKeyCollection, App.CurrentUser.Id);
            }
            else
            {
                MessageBox.Show("The key exist already.");
```

```
            }
        }

        private void CancelButton_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.GoBack();
        }
    }
}
```

D.2.13 Cryptbook.ImportKeyMaterial.xaml

```
<phone:PhoneApplicationPage
    x:Class="Cryptbook.ImportKeyMaterial"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-
->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="Manually Import Key
Material" Style="{StaticResource PhoneTextNormalStyle}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <TextBlock Text="Paste the private and public key material
(in xml form) into the below text boxes" Style="{StaticResource Pho-
neTextSmallStyle}" TextWrapping="Wrap"></TextBlock>
                <TextBlock Text=" "></TextBlock>
                <TextBlock Text="Key Material" Style="{StaticResource Pho-
neTextLargeStyle}"></TextBlock>
                <TextBox x:Name="KeyTextBox" Text=""
Height="472"></TextBox>
                <TextBlock Text=" "></TextBlock>

                <StackPanel Orientation="Horizontal">
                    <Button x:Name="CancelButton" Content="Cancel"
Width="217" Click="CancelButton_Click"></Button>
                    <Button x:Name="ImportButton" Content="Import"
Width="234" Click="ImportButton_Click"></Button>
```

```
                </StackPanel>
            </StackPanel>

        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

D.2.14 Cryptbook.LoginPage

```
namespace Cryptbook
{
    public partial class LoginPage : PhoneApplicationPage
    {
        public LoginPage()
        {
            InitializeComponent();

            this.Loaded += new RoutedEventHandler(LoginPage_Loaded);
        }

        private void LoginPage_Loaded(object sender, RoutedEventArgs e)
        {
            StartPleaseWaitAnimation();
            webBrowser1.Navigate(new Uri(FacebookHelper.GetOAuth2Endpoint));
        }

        private void webBrowser1_LoadCompleted(object sender, Sys-
tem.Windows.Navigation.NavigationEventArgs e)
        {
            //inital load?
            if
(!(e.Uri.AbsoluteUri.ToLower().StartsWith(FacebookHelper.RedirectURL))
                && (!(e.Uri.AbsoluteUri.ToLower() == "about:blank")))
            {
                webBrowser1.Visibility = System.Windows.Visibility.Visible;
                StopPleaseWaitAnimation();
            }

            //Access Token retrieved?
            if
(e.Uri.AbsoluteUri.ToLower().StartsWith(FacebookHelper.RedirectURL))
            {
                App.AccessToken =
e.Uri.OriginalString.Substring(e.Uri.AbsoluteUri.IndexOf("#access_token=") +
14);
                if (App.AccessToken != string.Empty)
                {
                    App.IsUserAuthenticated = true;
                    webBrowser1.Navigate(new Uri("about:blank"));
                    webBrowser1.Visibility = Sys-
tem.Windows.Visibility.Collapsed;

                    StartPleaseWaitAnimation();

                    LoadCurrentUser();
                }
            }
        }

        private void StartPleaseWaitAnimation()
        {
```

```
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Visible;
        }

        private void StopPleaseWaitAnimation()
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;
        }

        private void LoadCurrentUser()
        {
            FacebookProxy proxy = new FacebookProxy(App.AccessToken);
            proxy.GetUserInfo("me", GetUserInfo_Completed);
        }

        void GetUserInfo_Completed(DownloadStringCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("something unexpected happened: " + e.Result);
                return;
            }

            string result = e.Result;
            App.CurrentUser = DataContracts.Friend.DeserializeJSON(e.Result);


            //load the keys for this user
            App.PrivatePublicKeyCollection = AsymmetricKeyFileManag-
er.Load(App.CurrentUser.Id);
            //App.PrivatePublicKey = App.PrivatePublicKeyCollection;

            StopPleaseWaitAnimation();

             //Finally navigate to the main Application
            NavigationService.Navigate(new Uri("/MainPage.xaml",
UriKind.Relative));
        }

        private void webBrowser1_Navigating(object sender, NavigatingEventArgs
e)
        {
            webBrowser1.Visibility = System.Windows.Visibility.Collapsed;
            StartPleaseWaitAnimation();
        }

    }
}
```

D.2.15 Cryptbook.LoginPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="Cryptbook.LoginPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
    xmlns:my="clr-namespace:Cryptbook"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"

    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Canvas x:Name="LayoutRoot" Background="Transparent">

        <phone:WebBrowser Name="webBrowser1" Canvas.Left="40"  Canvas.Top="190"
Canvas.ZIndex="5"
                          Width="400" Height="467"
                          LoadCompleted="webBrowser1_LoadCompleted" Navi-
gating="webBrowser1_Navigating" />
        <toolkit:PerformanceProgressBar Canvas.Left="0" Canvas.Top="430"
Width="480" x:Name="PleaseWaitProgressBar" IsIndeterminate="True" Visibil-
ity="Visible"></toolkit:PerformanceProgressBar>




        <TextBlock Canvas.Left="60" Canvas.Top="80" x:Name="PageTitle"
Text="Project Cryptbook" FontWeight="Bold" FontFamily="Calibri" FontSize="48"
/>
       <!-- <TextBlock Canvas.Left="190" Canvas.Top="135" Text="...a client for
the"  FontSize="16" Style="{StaticResource PhoneTextSubtleStyle}"></TextBlock>-
->
        <TextBlock Canvas.Left="250" Canvas.Top="135" Text="Privacy Solution
Center"  FontSize="16" Style="{StaticResource PhoneTextSubtle-
Style}"></TextBlock>

        <TextBlock  x:Name="LogonText" Text="Logon"  Style="{StaticResource
PhoneTextNormalStyle}" Canvas.Top="400" Canvas.Left="200"/>

        <TextBlock Canvas.Left="205" Canvas.Top="680" Text="idea, design and
implementation"  FontSize="16" ></TextBlock>
        <TextBlock Canvas.Left="230" Canvas.Top="705" Text="by Johann Rehberger
(2011)"  FontSize="16"  Style="{StaticResource PhoneTextSubtleStyle}" />


    </Canvas>
</phone:PhoneApplicationPage>
```

D.2.16 Cryptbook.MainPage

```
namespace Cryptbook
{
    public partial class MainPage : PhoneApplicationPage
    {
        // Constructor
        public MainPage()
        {
            InitializeComponent();

            this.Loaded += new RoutedEventHandler(MainPage_Loaded);
        }
```

```
        // Load data for the ViewModel Items
        private void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            UserInfo.DataContext = App.CurrentUser;
        }


        private void LoadData()
        {
            if (App.PrivatePublicKey == null)
            {
                StatuxBox.Text = "You have no public/private key pair. Please
go to asymmetric keys and create one.";
            }

            WallUserControl.LoadWallPostsAsync("me", 20, DateTime.UtcNow + new
TimeSpan(1,0,0), false);

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);
            proxy.GetMyFriendsAsync(GetFriendsAsync_Completed);
        }

        protected override void OnNavi-
gatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            LoadData();

            base.OnNavigatedTo(e);
        }



        void GetFriendsAsync_Completed(DownloadStringCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("something unexpected happened");
                return;
            }

            string result = e.Result;
            FriendsListBox.DataContext = DataCon-
tracts.Friends.DeserializeJSON(result).Friendlist;
        }


        private void FriendsListBox_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs e)
        {
            Friend friend = FriendsListBox.SelectedItem as Friend;
            if (friend != null)
            {
                string wall = string.Format("/Wall.xaml?id={0}&name={1}",
                   HttpUtility.UrlEncode(friend.Id),
                   HttpUtility.UrlEncode(friend.Name));
                NavigationService.Navigate(new Uri(wall, UriKind.Relative));
            }

        }

        private void PostTextBox_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs  e)
        {
            string resource =
string.Format("/WallPostPage.xaml?id={0}&name={1}", "me", "me");
            NavigationService.Navigate(new Uri(resource, UriKind.Relative));
```

```
        }


        private void AsymKeyMenuItem_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs  e)
        {
            string wall =
string.Format("/AsymmetricKeyManager.xaml?id={0}&name={1}",
                  "me",
                  "me");
            NavigationService.Navigate(new Uri(wall, UriKind.Relative));
        }

        private void LogoutMenuItem_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs  e)
        {
            if (MessageBox.Show("Are you sure you want to logout?", "Logout",
MessageBoxButton.OKCancel) == MessageBoxResult.OK)
            {
                Logout();
            }
        }



        #region Logout
        public void Logout()
        {
            try
            {
                Uri logoutUrl = new
Uri(FacebookHelper.GetLogoutEndpoint(App.AccessToken));

                WebClient request = new WebClient();
                request.DownloadStringAsync(logoutUrl);
                request.DownloadStringCompleted += new DownloadStringCom-
pletedEventHandler(requestLogout_DownloadStringCompleted);
            }
            catch (Exception ex)
            {
                MessageBox.Show( ex.Message );
            }
        }

        void requestLogout_DownloadStringCompleted(object sender, Down-
loadStringCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("something unexpected happened");
                return;
            }

            string result = e.Result;

            App.CurrentUser = null;
            App.AccessToken = null;
            App.IsUserAuthenticated = false;
            App.Friendlist = null;

            FriendsListBox.DataContext = null;
            WallUserControl.WallPostListBox.DataContext = null;

            NavigationService.Navigate(new Uri("/LoginPage.xaml",
UriKind.Relative));
        }
```

```
        #endregion


        private void WallMenu_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs e)
        {
            PanoramaControl.DefaultItem = WallPanoramaItem;
        }

        private void FriendMenu_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs e)
        {
            PanoramaControl.DefaultItem = FriendsPanoramaItem;
        }

        private void PhoneApplicationPage_BackKeyPress(object sender, Sys-
tem.ComponentModel.CancelEventArgs e)
        {
            if (PanoramaControl.SelectedItem != MainMenuPanoramaItem)
            {
                PanoramaControl.DefaultItem = MainMenuPanoramaItem;
                e.Cancel = true;
            }
        }
    }
}
```

D.2.17 Cryptbook.MainPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="Cryptbook.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:controlsPrimitives="clr-
namespace:Microsoft.Phone.Controls.Primitives;assembly=Microsoft.Phone.Controls
"
    xmlns:my="clr-namespace:Cryptbook"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"

    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait"  Orientation="Portrait"
    shell:SystemTray.IsVisible="True" BackKey-
Press="PhoneApplicationPage_BackKeyPress">

    <phone:PhoneApplicationPage.Resources>
        <Style x:Key="PanoramaStyle1" TargetType="controls:Panorama">
            <Setter Property="ItemsPanel">
                <Setter.Value>
                    <ItemsPanelTemplate>
                        <controlsPrimitives:PanoramaPanel x:Name="panel"/>
                    </ItemsPanelTemplate>
```

```
                </Setter.Value>
            </Setter>
            <Setter Property="Foreground" Value="{StaticResource PhoneFore-
groundBrush}"/>
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="controls:Panorama">
                        <Grid>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="auto"/>
                                <RowDefinition Height="*"/>
                            </Grid.RowDefinitions>
                            <controlsPrimitives:PanningBackgroundLayer
x:Name="BackgroundLayer" HorizontalAlignment="Left" Grid.RowSpan="2">
                                <Border x:Name="background" Back-
ground="{TemplateBinding Background}" CacheMode="BitmapCache"/>
                            </controlsPrimitives:PanningBackgroundLayer>
                            <controlsPrimitives:PanningTitleLayer
x:Name="TitleLayer" CacheMode="BitmapCache" ContentTemplate="{TemplateBinding
TitleTemplate}" Content="{TemplateBinding Title}" FontSize="50" FontFami-
ly="Calibri" HorizontalAlignment="Left" Margin="10,-15,0,9" Grid.Row="0"/>
                            <controlsPrimitives:PanningLayer
x:Name="ItemsLayer" HorizontalAlignment="Left" Grid.Row="1" FontSize="96" >
                                <ItemsPresenter x:Name="items"/>
                            </controlsPrimitives:PanningLayer>
                        </Grid>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </phone:PhoneApplicationPage.Resources>

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent" >

        <!--Panorama control-->
        <controls:Panorama x:Name="PanoramaControl" Title="Project Cryptbook"
Style="{StaticResource PanoramaStyle1}" >
            <controls:Panorama.Background>
                <ImageBrush ImageSource=""/>
            </controls:Panorama.Background>

            <controls:PanoramaItem x:Name="MainMenuPanoramaItem" >
                <controls:PanoramaItem.Header>
                    <TextBlock FontSize="44" ></TextBlock>
                </controls:PanoramaItem.Header>

                <StackPanel>
                        <StackPanel x:Name="UserInfo" Orientation="Horizontal"
>
                        <Image Source="{Binding Picture}" Width="128"
Height="128" Margin="12,0,0,0" />
                        <TextBlock Text="{Binding Name}" TextWrapping="Wrap"
Style="{StaticResource PhoneTextLargeStyle}" Margin="30,0,12,0" />

                    </StackPanel>
                    <toolkit:Separator Margin="0,20,0,0"></toolkit:Separator>
                    <StackPanel Width="432" Height="381" Margin="0,30,0,0">
                        <TextBlock Text="my wall" x:Name="WallMenu" TextWrap-
ping="Wrap" Style="{StaticResource PhoneTextExtraLargeStyle}"
Tap="WallMenu_Tap" />
                        <TextBlock Text="friends" x:Name="FriendMenu" TextWrap-
ping="Wrap" Style="{StaticResource PhoneTextExtraLargeStyle}"
Tap="FriendMenu_Tap" />
```

```
                        <TextBlock Text="keys" x:Name="AsymKeyMenuItem"
TextWrapping="Wrap" Style="{StaticResource PhoneTextExtraLargeStyle}"
Tap="AsymKeyMenuItem_Tap" />
                        <TextBlock Text="logout" x:Name="LogoutMenuItem"
TextWrapping="Wrap" Style="{StaticResource PhoneTextExtraLargeStyle}"
Tap="LogoutMenuItem_Tap" />
                    </StackPanel>
                    <TextBlock Text="" x:Name="StatuxBox" TextWrapping="Wrap"
Style="{StaticResource PhoneTextNormalStyle}" Height="99" Width="411" />
                </StackPanel>

            </controls:PanoramaItem>

            <controls:PanoramaItem x:Name="WallPanoramaItem"  >
                <controls:PanoramaItem.Header>
                    <TextBlock FontSize="44">my wall</TextBlock>
                </controls:PanoramaItem.Header>
                <StackPanel Margin="0,-20,0,0">

                    <TextBox x:Name="PostTextBox" Text="Share your thoughts"
FontSize="24" Height="70" Margin="28,0,45,15" TextWrapping="NoWrap" Width="392"
Tap="PostTextBox_Tap" />

                    <my:WallUserControl x:Name="WallUserControl" Height="562">
                    </my:WallUserControl>
                </StackPanel>
            </controls:PanoramaItem>

            <controls:PanoramaItem x:Name="FriendsPanoramaItem">
                <controls:PanoramaItem.Header>
                    <TextBlock FontSize="44">friends</TextBlock>
                </controls:PanoramaItem.Header>

                <ListBox x:Name="FriendsListBox" Margin="0,0,-12,0" Items-
Source="{Binding}"  Tap="FriendsListBox_Tap">
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                            <StackPanel Margin="0,0,0,17">
                                <TextBlock Text="{Binding Name}" TextWrap-
ping="NoWrap" Margin="12,0,0,0" Style="{StaticResource PhoneTextLargeStyle}"
/>
                                <Image Source="{Binding Picture}" Height="64"
Width="64" HorizontalAlignment="Left" />
                            </StackPanel>
                        </DataTemplate>
                    </ListBox.ItemTemplate>
                </ListBox>
            </controls:PanoramaItem>
        </controls:Panorama>

    </Grid>

</phone:PhoneApplicationPage>
```

D.2.18 Cryptbook.ViewPostAndComments

```
namespace Cryptbook
{
    public partial class ViewPostAndComments : PhoneApplicationPage
    {

        private string postId;
       // private string facebookName;
```

```
        List<WallPost> wallposts;


        public ViewPostAndComments()
        {
            InitializeComponent();
            WallPostListBox.DataContext = wallposts;
        }

        protected override void OnNavi-
gatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            postId = NavigationContext.QueryString.Where(item => item.Key ==
"id").FirstOrDefault().Value;
          // facebookName= NavigationContext.QueryString.Where(item =>
item.Key == "name").FirstOrDefault().Value;
            PageTitle.Text = "Post and Comments";// facebookName.ToLower() +
"'s wall";

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);
            proxy.GetWallAsync(postId, DateTime.Now + new TimeSpan(1,0,0), Get-
WallAsync_Completed);


            //WallUserControl.LoadWallPostsAsync(facebookId, 20,
DateTime.UtcNow, false);

            base.OnNavigatedTo(e);
        }

        void GetWallAsync_Completed(DownloadStringCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("something unexpected happened");
                return;
            }

            string result = e.Result;

            var json = DataContracts.WallPosts.DeserializeJSON(result);


        }

        protected override void OnBackKey-
Press(System.ComponentModel.CancelEventArgs e)
        {
            base.OnBackKeyPress(e);
        }

        private void PostTextBox_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs  e)
        {
            string resource =
string.Format("/WallPostPage.xaml?id={0}&name={1}", postId, "tbd");
            NavigationService.Navigate(new Uri(resource, UriKind.Relative));
        }

    }
}
```

D.2.19 Cryptbook.ViewPostsAndComments.xaml

```
<phone:PhoneApplicationPage
    x:Class="Cryptbook.ViewPostAndComments"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:my="clr-namespace:Cryptbook"
     xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="Project Cryptbook"
Style="{StaticResource PhoneTextLargeStyle}"/>
            <TextBlock x:Name="PageTitle" Text="" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextExtraLargeStyle}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <StackPanel>
                <TextBox x:Name="PostTextBox" Text="add a comment" Font-
Weight="Thin" FontSize="24" Height="70" Margin="28,10,45,15" TextWrap-
ping="NoWrap" Width="392"/>

                <StackPanel Margin="12,0,12,0" x:Name="AddPost" Visibil-
ity="Visible" >
                    <ListBox x:Name="WallPostListBox" Margin="0,0,0,0" Items-
Source="{Binding}" Visibility="Visible" Height="584" Width="440" >
                        <ListBox.ItemTemplate>
                            <DataTemplate>
                                <StackPanel>
                                    <StackPanel Orientation="Horizontal">
                                        <Image Source="{Binding From.Picture}"
Height="40" Width="40" HorizontalAlignment="Left" Margin="0,0,0,0" />

                                        <StackPanel Orientation="Vertical">
                                            <TextBlock Text="{Binding
From.Name}"  x:Name="PostersName" Width="200" TextWrapping="Wrap" Mar-
gin="10,2,0,0" Style="{StaticResource PhoneTextNormalStyle}"/>
                                            <TextBlock Text="{Binding Get-
FriendlyCreatedTime}"  x:Name="CreatedTime" Width="200" TextWrapping="Wrap"
Margin="10,2,0,0" Style="{StaticResource PhoneTextSmallStyle}"/>
                                        </StackPanel>
                                        <Button Content="comment" Horizonta-
lAlignment="Right" Width="110" x:Name="CommentButton" Margin="5,0,-10,0"
FontFamily="Calibri" FontSize="16"></Button>
```

```
                                    <TextBlock Text="{Binding From.Id}"
x:Name="PostersID"  Visibility="Collapsed" Width="380" TextWrapping="Wrap" Mar-
gin="12,-6,12,0"   />
                                    <TextBlock Text="{Binding Id}"
x:Name="PostID" Visibility="Collapsed" Width="380" TextWrapping="Wrap" Mar-
gin="12,-6,12,0" />
                                </StackPanel>

                                <TextBlock Text="{Binding Message}"
Width="380" TextWrapping="Wrap" Margin="12,-6,12,0"  Style="{StaticResource
PhoneTextNormalStyle}"/>
                                <TextBlock Text="{Binding Name}"
Width="380" TextWrapping="Wrap" Margin="12,-6,12,0"  Style="{StaticResource
PhoneTextNormalStyle}"/>
                                <TextBlock Text="{Binding Description}"
Width="380" TextWrapping="Wrap" Margin="12,-6,12,0" Style="{StaticResource Pho-
neTextNormalStyle}"/>

                                <ListBox x:Name="Comment" Margin="0,0,0,0"
ItemsSource="{Binding CommentsList}" Visibility="Visible" Height="410"
Width="440">
                                    <ListBox.ItemTemplate>
                                        <DataTemplate>
                                            <StackPanel>
                                                <StackPanel Orienta-
tion="Horizontal">
                                                    <Image Source="{Binding
CommentsData.From.Picture}" Height="40" Width="40" HorizontalAlignment="Left"
Margin="0,0,0,0" />

                                                    <StackPanel Orienta-
tion="Vertical">
                                                        <TextBlock
Text="{Binding CommentsData.From.Name}"  x:Name="CommentPostersName"
Width="200" TextWrapping="Wrap" Margin="10,2,0,0" Style="{StaticResource Pho-
neTextNormalStyle}"/>
                                                        <TextBlock
Text="{Binding CommentsData.GetFriendlyCreatedTime}"
x:Name="CommentCreatedTime" Width="200" TextWrapping="Wrap" Margin="10,2,0,0"
Style="{StaticResource PhoneTextSmallStyle}"/>
                                                    </StackPanel>

                                                    <TextBlock
Text="{Binding CommentsData.From.Id}" x:Name="CommentPosterID"  Visibil-
ity="Collapsed" Width="380" TextWrapping="Wrap" Margin="12,-6,12,0"   />
                                                    <TextBlock
Text="{Binding CommentsData.Id}"      x:Name="CommentID" Visibility="Collapsed"
Width="380" TextWrapping="Wrap" Margin="12,-6,12,0" />
                                                </StackPanel>

                                                <TextBlock Text="{Binding
CommentsData.Message}" Width="380" TextWrapping="Wrap" Margin="12,-6,12,0"
Style="{StaticResource PhoneTextNormalStyle}"/>
                                                <TextBlock Text="{Binding
CommentsData.Name}" Width="380" TextWrapping="Wrap" Margin="12,-6,12,0"
Style="{StaticResource PhoneTextNormalStyle}"/>
                                                <TextBlock Text="{Binding
CommentsData.Description}" Width="380" TextWrapping="Wrap" Margin="12,-6,12,0"
Style="{StaticResource PhoneTextNormalStyle}"/>
                                                <TextBlock  Mar-
gin="0,0,0,20" Text=""></TextBlock>
                                            </StackPanel>
                                        </DataTemplate>
                                    </ListBox.ItemTemplate>
                                </ListBox>
```

```
                                    <TextBlock  Margin="0,0,0,20"
Text=""></TextBlock>
                                </StackPanel>
                            </DataTemplate>
                        </ListBox.ItemTemplate>
                    </ListBox>
                    <toolkit:PerformanceProgressBar
x:Name="PleaseWaitProgressBar" IsIndeterminate="True" Visibil-
ity="Collapsed"></toolkit:PerformanceProgressBar>
                </StackPanel>
            </StackPanel>
        </Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

D.2.20 Cryptbook.Wall

```
namespace Cryptbook
{
    public partial class Wall : PhoneApplicationPage
    {

        private string facebookId;
        private string facebookName;

        public Wall()
        {
            InitializeComponent();
        }

        protected override void OnNavi-
gatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            facebookId = NavigationContext.QueryString.Where(item => item.Key
== "id").FirstOrDefault().Value;
            facebookName= NavigationContext.QueryString.Where(item => item.Key
== "name").FirstOrDefault().Value;
            PageTitle.Text = facebookName.ToLower() + "'s wall";

            WallUserControl.LoadWallPostsAsync(facebookId, 20, DateTime.UtcNow
+ new TimeSpan(1,0,0), false);
            base.OnNavigatedTo(e);
        }

        protected override void OnBackKey-
Press(System.ComponentModel.CancelEventArgs e)
        {
            base.OnBackKeyPress(e);
        }

        private void PostTextBox_Tap(object sender, Sys-
tem.Windows.Input.GestureEventArgs  e)
        {
            string resource =
string.Format("/WallPostPage.xaml?id={0}&name={1}", facebookId, facebookName);
            NavigationService.Navigate(new Uri(resource, UriKind.Relative));
        }

    }
}
```

D.2.21 Cryptbook.Wall.xaml

```
<phone:PhoneApplicationPage
    x:Class="Cryptbook.Wall"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:my="clr-namespace:Cryptbook"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,18">
            <TextBlock x:Name="ApplicationTitle" Text="Project Cryptbook"
Style="{StaticResource PhoneTextLargeStyle}"/>
            <TextBlock x:Name="PageTitle" Text="" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextExtraLargeStyle}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,0,0">
            <StackPanel>
                <TextBox x:Name="PostTextBox" Text="Post something" Font-
Weight="Thin" FontSize="24" Height="70" Margin="28,0,45,15" TextWrap-
ping="NoWrap" Width="392" Tap="PostTextBox_Tap" />

                <my:WallUserControl x:Name="WallUserControl" Height="600">
                </my:WallUserControl>
            </StackPanel>
        </Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

D.2.22 Cryptbook.WallPost

```
namespace Cryptbook
{
    public partial class WallPostPage : PhoneApplicationPage
    {
        public WallPostPage()
        {
            InitializeComponent();
        }


        private string facebookId;
        private string facebookName;
```

```
        private string objectId;

        private const string DefaultPublicKeyLoadingText = "accessing public
key(s)...";

        private PostMode postMode = PostMode.Wall;
        private enum PostMode
        {
            Wall,
            Comment
        }


        protected override void OnNavi-
gatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            facebookId   = NavigationContext.QueryString.Where(item => item.Key
== "id").FirstOrDefault().Value;
            facebookName = NavigationContext.QueryString.Where(item => item.Key
== "name").FirstOrDefault().Value;

            KeyValuePair<string, string> mode = NavigationCon-
text.QueryString.Where(item => item.Key == "mode").FirstOrDefault();
            if ((mode.Value == null) || (mode.Value.ToLower() == "wall"))
            {
                postMode = PostMode.Wall;
                TitlePost.Text = "Share your thoughts";
            }
            else if (mode.Value.ToLower() == "comment")
            {
                postMode = PostMode.Comment;
                objectId = NavigationContext.QueryString.Where(item => item.Key
== "objectId").FirstOrDefault().Value;
                TitlePost.Text = "Share your comment";
            }

            //reinitialize the public keys
            recipientKeys = new List<AsymmetricKey>();

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);
            proxy.GetPublicKey("me", GetPublicKeyCallback, null);


            FacebookProxy[] proxies = new Facebook-
Proxy[App.SelectedRecipients.Count];
            int i = 0;
            foreach (DataContracts.Friend recipient in App.SelectedRecipients)
            {
                proxies[i] = new FacebookProxy(App.AccessToken);

                StateObject state = new StateObject();
                state.FacebookUser = recipient;
                proxies[i].GetPublicKey(recipient.Id, GetPublicKeyCallback,
state);
                i++;
            }

            //proxy.GetPublicKey(facebookId, GetPublicKeyCallback, null);

            base.OnNavigatedTo(e);
        }


        private void PostButton_Click(object sender, RoutedEventArgs e)
        {
```

```
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Visible;

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);

            if (postMode == PostMode.Wall)
            {
                proxy.PostWall(facebookId, MessageTextBox.Text, Wall-
Post_UploadStringCompleted);
            }

            if (postMode == PostMode.Comment)
            {
                proxy.PostComment(objectId, MessageTextBox.Text, Wall-
Post_UploadStringCompleted);
            }
        }



        //we just pick the very first one we find
        List<AsymmetricKey> recipientKeys = new List<AsymmetricKey>();

        private void GetPublicKeyCallback(DownloadStringCompletedEventArgs e)
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;


            if (e.Error != null)
            {
                string message = "Error reading public key: " +
e.Error.Message;

                var user = (e.UserState as StateObject).FacebookUser;
                if (user != null)
                {
                    message = "Error reading " + user.Name + "'s public key: "
+ e.Error.Message;
                }

                MessageBox.Show(message);
            }
            else
            {
                //get the response
                string objectID = e.Result;

                try
                {
                    AsymmetricKey key = FacebookHelp-
er.ExtractPublicKeyFromJsonResponse(e.Result);
                    recipientKeys.Add(key);

                    lock (SelectedKey)
                    {
                        if (SelectedKey.Text == DefaultPublicKeyLoadingText)
                        {
                            SelectedKey.Text = string.Empty;
                        }

                        var user = (e.UserState as StateObject).FacebookUser;
                        if (user != null)
                        {
                            if (SelectedKey.Text.Length > 0)
                            {
```

```
                                SelectedKey.Text += ", " + user.Name;
                            }
                            else
                            {
                                SelectedKey.Text = user.Name;
                            }
                        }
                        else
                        {
                            if (SelectedKey.Text.Length > 0)
                            {
                                SelectedKey.Text += ", me";
                            }
                            else
                            {
                                SelectedKey.Text = "me";
                            }
                        }
                    }
                }
                catch (Exception ex)
                {
                    string message = "Error reading public key: " + ex.Message;

                    var user = (e.UserState as StateObject).FacebookUser;
                    if (user != null)
                    {
                        message = "Error reading " + user.Name + "'s public
key: " + ex.Message;
                    }

                    MessageBox.Show(message);
                }
                finally
                {
                    PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;
                }
            }
        }

        void WallPost_UploadStringCompleted(UploadStringCompletedEventArgs e)
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;

            if (e.Error != null)
            {
                MessageBox.Show("Error occured: " + e.Error.Message);
            }
            else
            {
                //get the response
                string objectID = e.Result;
                NavigationService.GoBack();
            }
        }

        private void SelectEncryptionKeys_Click(object sender, RoutedEventArgs
e)
        {
            SelectedKey.Text = DefaultPublicKeyLoadingText;
            NavigationService.Navigate(new Uri("/FriendSelector.xaml",
UriKind.Relative));
        }
```

```
        private void Encrypt_Click(object sender, RoutedEventArgs e)
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Visible;

            string clearText = MessageTextBox.Text;


            if (App.PrivatePublicKey == null)
            {
                MessageBox.Show("You don't have a private key to sign the mes-
sage. Please create a key pair.");
                return;

            }

            MessageTextBox.Text = CryptoHelper.EncryptAsymmetric(recipientKeys,
App.PrivatePublicKey, clearText);
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;
        }

    }
}
```

D.2.23 Cryptbook.WallPost.xaml

```
<phone:PhoneApplicationPage
    x:Class="Cryptbook.WallPostPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True" xmlns:my="clr-
namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
>

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <StackPanel>
            <StackPanel x:Name="UpperPanel">
                <TextBlock x:Name="TitlePost" Text="Share your thoughts:" Font-
Size="26"></TextBlock>
                <TextBox x:Name="MessageTextBox" Height="444" AcceptsRe-
turn="True" TextWrapping="Wrap" FontSize="24"></TextBox>
                <Button Visibility="Visible"  x:Name="SelectEncryptionKeys"
Content="Select Recipients" Click="SelectEncryptionKeys_Click"></Button>

                <TextBlock Text="Encrypt message for" TextAlignment="Center"
FontSize="26"></TextBlock>
                <TextBlock x:Name="SelectedKey" Text="accessing public
key(s)..." TextAlignment="Center" FontSize="26"></TextBlock>
```

```
                <Button x:Name="Encrypt"  Content="Encrypt"
Click="Encrypt_Click"></Button>
                <Button x:Name="PostButton"  Content="Post"
Click="PostButton_Click"></Button>
            </StackPanel>
        </StackPanel>
        <toolkit:PerformanceProgressBar x:Name="PleaseWaitProgressBar" IsInde-
terminate="True" Visibility="Collapsed"></toolkit:PerformanceProgressBar>
    </Grid>

</phone:PhoneApplicationPage>
```

D.2.24 Cryptbook.WallPostPage

```
namespace Cryptbook
{
    public partial class WallPostPage : PhoneApplicationPage
    {
        public WallPostPage()
        {
            InitializeComponent();
        }


        private string facebookId;
        private string facebookName;
        private string objectId;

        private const string DefaultPublicKeyLoadingText = "accessing public
key(s)...";

        private PostMode postMode = PostMode.Wall;
        private enum PostMode
        {
            Wall,
            Comment
        }


        protected override void OnNavi-
gatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            facebookId   = NavigationContext.QueryString.Where(item => item.Key
== "id").FirstOrDefault().Value;
            facebookName = NavigationContext.QueryString.Where(item => item.Key
== "name").FirstOrDefault().Value;

            KeyValuePair<string, string> mode = NavigationCon-
text.QueryString.Where(item => item.Key == "mode").FirstOrDefault();
            if ((mode.Value == null) || (mode.Value.ToLower() == "wall"))
            {
                postMode = PostMode.Wall;
                TitlePost.Text = "Share your thoughts";
            }
            else if (mode.Value.ToLower() == "comment")
            {
                postMode = PostMode.Comment;
                objectId = NavigationContext.QueryString.Where(item => item.Key
== "objectId").FirstOrDefault().Value;
                TitlePost.Text = "Share your comment";
            }

            //reinitialize the public keys
```

```
            recipientKeys = new List<AsymmetricKey>();

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);
            proxy.GetPublicKey("me", GetPublicKeyCallback, null);


            FacebookProxy[] proxies = new Facebook-
Proxy[App.SelectedRecipients.Count];
            int i = 0;
            foreach (DataContracts.Friend recipient in App.SelectedRecipients)
            {
                proxies[i] = new FacebookProxy(App.AccessToken);

                StateObject state = new StateObject();
                state.FacebookUser = recipient;
                proxies[i].GetPublicKey(recipient.Id, GetPublicKeyCallback,
state);
                i++;
            }

            //proxy.GetPublicKey(facebookId, GetPublicKeyCallback, null);

            base.OnNavigatedTo(e);
        }


        private void PostButton_Click(object sender, RoutedEventArgs e)
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Visible;

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);

            if (postMode == PostMode.Wall)
            {
                proxy.PostWall(facebookId, MessageTextBox.Text, Wall-
Post_UploadStringCompleted);
            }

            if (postMode == PostMode.Comment)
            {
                proxy.PostComment(objectId, MessageTextBox.Text, Wall-
Post_UploadStringCompleted);
            }
        }



        //we just pick the very first one we find
        List<AsymmetricKey> recipientKeys = new List<AsymmetricKey>();

        private void GetPublicKeyCallback(DownloadStringCompletedEventArgs e)
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;


            if (e.Error != null)
            {
                string message = "Error reading public key: " +
e.Error.Message;

                var user = (e.UserState as StateObject).FacebookUser;
                if (user != null)
                {
```

```
                    message = "Error reading " + user.Name + "'s public key: "
+ e.Error.Message;
                }

                MessageBox.Show(message);
            }
            else
            {
                //get the response
                string objectID = e.Result;

                try
                {
                    AsymmetricKey key = FacebookHelp-
er.ExtractPublicKeyFromJsonResponse(e.Result);
                    recipientKeys.Add(key);

                    lock (SelectedKey)
                    {
                        if (SelectedKey.Text == DefaultPublicKeyLoadingText)
                        {
                            SelectedKey.Text = string.Empty;
                        }

                        var user = (e.UserState as StateObject).FacebookUser;
                        if (user != null)
                        {
                            if (SelectedKey.Text.Length > 0)
                            {
                                SelectedKey.Text += ", " + user.Name;
                            }
                            else
                            {
                                SelectedKey.Text = user.Name;
                            }
                        }
                        else
                        {
                            if (SelectedKey.Text.Length > 0)
                            {
                                SelectedKey.Text += ", me";
                            }
                            else
                            {
                                SelectedKey.Text = "me";
                            }
                        }
                    }
                }
                catch (Exception ex)
                {
                    string message = "Error reading public key: " + ex.Message;

                    var user = (e.UserState as StateObject).FacebookUser;
                    if (user != null)
                    {
                        message = "Error reading " + user.Name + "'s public
key: " + ex.Message;
                    }

                    MessageBox.Show(message);
                }
                finally
                {
                    PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;
```

```
                }
            }
        }

        void WallPost_UploadStringCompleted(UploadStringCompletedEventArgs e)
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;

            if (e.Error != null)
            {
                MessageBox.Show("Error occured: " + e.Error.Message);
            }
            else
            {
                //get the response
                string objectID = e.Result;
                NavigationService.GoBack();
            }
        }

        private void SelectEncryptionKeys_Click(object sender, RoutedEventArgs
e)
        {
            SelectedKey.Text = DefaultPublicKeyLoadingText;
            NavigationService.Navigate(new Uri("/FriendSelector.xaml",
UriKind.Relative));
        }

        private void Encrypt_Click(object sender, RoutedEventArgs e)
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Visible;

            string clearText = MessageTextBox.Text;


            if (App.PrivatePublicKey == null)
            {
                MessageBox.Show("You don't have a private key to sign the mes-
sage. Please create a key pair.");
                return;

            }

            MessageTextBox.Text = CryptoHelper.EncryptAsymmetric(recipientKeys,
App.PrivatePublicKey, clearText);
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;
        }

    }
}
```

D.2.25 Cryptbook.WallPostPage.xaml

```
<phone:PhoneApplicationPage
    x:Class="Cryptbook.WallPostPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True" xmlns:my="clr-
namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
>

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <StackPanel>
            <StackPanel x:Name="UpperPanel">
                <TextBlock x:Name="TitlePost" Text="Share your thoughts:" Font-
Size="26"></TextBlock>
                <TextBox x:Name="MessageTextBox" Height="444" AcceptsRe-
turn="True" TextWrapping="Wrap" FontSize="24"></TextBox>
                <Button Visibility="Visible"  x:Name="SelectEncryptionKeys"
Content="Select Recipients" Click="SelectEncryptionKeys_Click"></Button>

                <TextBlock Text="Encrypt message for" TextAlignment="Center"
FontSize="26"></TextBlock>
                <TextBlock x:Name="SelectedKey" Text="accessing public
key(s)..." TextAlignment="Center" FontSize="26"></TextBlock>

                <Button x:Name="Encrypt"  Content="Encrypt"
Click="Encrypt_Click"></Button>
                <Button x:Name="PostButton"  Content="Post"
Click="PostButton_Click"></Button>
            </StackPanel>
        </StackPanel>
        <toolkit:PerformanceProgressBar x:Name="PleaseWaitProgressBar" IsInde-
terminate="True" Visibility="Collapsed"></toolkit:PerformanceProgressBar>
    </Grid>
</phone:PhoneApplicationPage>
```

D.2.26 Cryptbook.WallUserControl

```
namespace Cryptbook
{
    public partial class WallUserControl : UserControl
    {
        public WallUserControl()
        {
            InitializeComponent();
            WallPostListBox.DataContext = userWall;
            this.Loaded += new RoutedEventHandler(Control_Loaded);
        }

        private string facebookId;
        private ObservableCollection<WallPost> userWall = new ObservableCollec-
tion<WallPost>();

        #region LoadWall
        public void LoadWallPostsAsync(string facebookId, int resultLimit,
DateTime until, bool append)
        {
            this.facebookId = facebookId;
```

```
            FacebookProxy proxy = new FacebookProxy(App.AccessToken);

            if (append)
            {
                proxy.GetWallAsync(facebookId, until, GetWallPostsAsyncAp-
pend_Completed);
            }
            else
            {
                proxy.GetWallAsync(facebookId, until, GetWallPostsAsyn-
cNew_Completed);
            }
        }

        void GetWallPostsAsyncNew_Completed(DownloadStringCompletedEventArgs e)
        {
            ProcessGetWallPostAsync(e, false);
        }


        void GetWallPostsAsyncAppend_Completed(DownloadStringCompletedEventArgs
e)
        {
            ProcessGetWallPostAsync(e, true);
        }

        void ProcessGetWallPostAsync(DownloadStringCompletedEventArgs e, bool
append)
        {
            if (e.Error != null)
            {
                MessageBox.Show("something unexpected happened");
                return;
            }

            string result = e.Result;

            var json = DataContracts.WallPosts.DeserializeJSON(result);

            if (append)
            {
                foreach (var post in json.Wallposts)
                {
                    userWall.Add(post);
                }
            }
            else
            {
                userWall = json.Wallposts;
            }

            //load the images for each post
            userWall.Distinct(new WallPostFriendCompar-
er()).ToList().ForEach(wallpost =>
            {
                FacebookProxy proxy = new FacebookProxy(App.AccessToken);
                proxy.GetPictureOfUser(wallpost.From.Id, GetPictureOfUs-
er_Completed);
            }
            );


            //currently we always only have one decryption key stored
            //the DecryptAsymmmetric API is designed to handle multiple
            List<AsymmetricKey> privateKeys = new List<AsymmetricKey>();
            privateKeys.Add(App.PrivatePublicKey);
```

```
            WallPostListBox.DataContext = userWall;

            //display the wall post
            foreach (WallPost post in json.Wallposts)
            {
                //currently we only support posts that have the message proper-
ty set
                if ((post.ApplicationName == null) || (post.Message == null))
                {
                    continue;
                }

                //also must be a cryptbook post
                if (post.ApplicationName.Id != FacebookHelper.AppID)
                {
                    continue;
                }

                GetPublicKeyForTryToDecrypt(post, privateKeys);

                if (post.CommentsList.CommentsData != null)
                {
                    foreach (var comment in post.CommentsList.CommentsData)
                    {
                        GetPublicKeyForTryToDecrypt(comment, privateKeys);
                    }
                }
            }

            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Collapsed;
        }



        private void GetPublicKeyForTryToDecrypt(WallPost post,
List<AsymmetricKey> privateKeys)
        {

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);
            StateObject userdata = new StateObject();

            //do a quick sanity check to see if we should even try to decrypt
            if (!post.Message.StartsWith("#CB|"))
                return;

            if (post.Message.Split('|').Length != 6)
                return;

            userdata.Type = MessageType.Wallpost;
            userdata.WallPost = post;
            userdata.PrivateKeys = privateKeys;
            proxy.GetPublicKey(post.From.Id, GetPublicKeyForDecryptionAndDe-
crypt_Completed, userdata);
            return;
        }

        private void GetPublicKeyForTryToDecrypt(WallPost.Comment comment,
List<AsymmetricKey> privateKeys)
        {

            FacebookProxy proxy = new FacebookProxy(App.AccessToken);
            StateObject userdata = new StateObject();

            //do a quick sanity check to see if we should even try to decrypt
```

```
            if (!comment.Message.StartsWith("#CB|"))
                return;

            if (comment.Message.Split('|').Length != 6)
                return;

            userdata.Type = MessageType.Comment;
            userdata.Comment = comment;
            userdata.PrivateKeys = privateKeys;
            proxy.GetPublicKey(comment.From.Id, GetPublicKeyForDecryptionAndDe-
crypt_Completed, userdata);
        }


        void GetPublicKeyForDecryptionAndDe-
crypt_Completed(DownloadStringCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("something unexpected happened");
                return;
            }

            AsymmetricKey publicKey = FacebookHelp-
er.ExtractPublicKeyFromJsonResponse(e.Result);
            StateObject s = e.UserState as StateObject;
            if (s != null)
            {
                   string orig = string.Empty;
                    //decrypt and update the message on the wall
                    try
                    {

                        if (s.Type == MessageType.Wallpost)
                        {
                            orig = s.WallPost.Message;

                            string decrypted = CryptoHelp-
er.DecryptAsymmetric(s.PrivateKeys, publicKey, s.WallPost.Message);
                            s.WallPost.Message = decrypted;
                            s.WallPost.IsEncrypted = true;
                        }
                        else //Comment
                        {
                            orig = s.Comment.Message;
                            string decrypted = CryptoHelp-
er.DecryptAsymmetric(s.PrivateKeys, publicKey, s.Comment.Message);
                            s.Comment.Message = decrypted;
                            s.Comment.IsEncrypted = true;
                        }

                    }
                    catch (Exception ex)
                    {
                        if (s.Type == MessageType.Wallpost)
                        {
                            s.WallPost.Message = "Unable to decrypt. Original
message: " + orig; // +ex.ToString()
                        }
                        else
                        {
                            s.Comment.Message = "Unable to decrypt. Original
message: " + orig; // +ex.ToString()
                        }
                    }
```

```
            }
        }

        void GetPictureOfUser_Completed(DownloadStringCompletedEventArgs e)
        {
            if (e.Error != null)
            {
                MessageBox.Show("something unexpected happened");
                return;
            }

            string result = e.Result;

            Friend friend = DataContracts.Friend.DeserializeJSON(result);
            userWall.Where(wallpost => wallpost.From.Id == friend.Id).ToList()
                .ForEach( friendpost => friendpost.From.Picture =
friend.Picture);

        }

    #endregion

        private void WallPostListBox_DoubleTap(object sender, Sys-
tem.Windows.Input.GestureEventArgs e)
        {
            LoadWallPostsAsync(facebookId, 20, DateTime.UtcNow + new
TimeSpan(1,0,0), false);
        }

        private void CommentButton_Click(object sender, RoutedEventArgs e)
        {
            string objectId = (((e.OriginalSource as Button).Parent as Stack-
Panel).FindName("PostID") as TextBlock).Text;
            string userid = (((e.OriginalSource as Button).Parent as StackPan-
el).FindName("PostersID") as TextBlock).Text;
            string username = (((e.OriginalSource as Button).Parent as Stack-
Panel).FindName("PostersName") as TextBlock).Text;

            string resource =
string.Format("/WallPostPage.xaml?mode=comment&id={0}&name={1}&objectId={2}",
userid, username, objectId);

            //leverage the NavigationService of the host page,
            //could be done better via an event I assume
            (Application.Current.RootVisual as PhoneApplication-
Frame).Navigate(new Uri(resource, UriKind.Relative));
        }

        private void ViewCommentsButton_Click(object sender, RoutedEventArgs e)
        {
            ListBox commentsbox = (((e.OriginalSource as HyperlinkBut-
ton).Parent as StackPanel).FindName("CommentListBox") as ListBox);

            //toggle visibilty
            if (commentsbox.Visibility == System.Windows.Visibility.Collapsed)
            {
                commentsbox.Visibility = System.Windows.Visibility.Visible;
            }
            else
            {
                commentsbox.Visibility = System.Windows.Visibility.Collapsed;
            }

        }

        private void LoadAdditionalData()
```

```
        {
            PleaseWaitProgressBar.Visibility = Sys-
tem.Windows.Visibility.Visible;

            WallPost oldestPost = userWall.OrderBy(post =>
post.CreatedTime).FirstOrDefault();
            LoadWallPostsAsync(facebookId, 20,
DateTime.Parse(oldestPost.CreatedTime), true);
        }


        #region End List Scroll Detection

        //Listbox Scrolling end of list detection
        //http://blogs.msdn.com/b/slmperf/archive/2011/06/30/windows-phone-
mango-change-listbox-how-to-detect-compression-end-of-scroll-states.aspx

        private ScrollBar sb = null;
        private ScrollViewer sv = null;
        private bool alreadyHookedScrollEvents = false;
        private void Control_Loaded(object sender, RoutedEventArgs e)
        {
            if (alreadyHookedScrollEvents)
                return;

            alreadyHookedScrollEvents = true;

            sb = (ScrollBar)FindElementRecursive(WallPostListBox,
typeof(ScrollBar));
            sv = (ScrollViewer)FindElementRecursive(WallPostListBox,
typeof(ScrollViewer));

            if (sv != null)
            {
                // Visual States are always on the first child of the control
template
                FrameworkElement element = VisualTreeHelper.GetChild(sv, 0) as
FrameworkElement;
                if (element != null)
                {
                    VisualStateGroup vgroup = FindVisualState(element, "Verti-
calCompression");
                    if (vgroup != null)
                    {
                        vgroup.CurrentStateChanging += new
EventHandler<VisualStateChangedEventArgs>(vgroup_CurrentStateChanging);
                    }
                }
            }
        }



        private UIElement FindElementRecursive(FrameworkElement parent, Type
targetType)
        {
            int childCount = VisualTreeHelper.GetChildrenCount(parent);
            UIElement returnElement = null;
            if (childCount > 0)
            {
                for (int i = 0; i < childCount; i++)
                {
                    Object element = VisualTreeHelper.GetChild(parent, i);
                    if (element.GetType() == targetType)
                    {
                        return element as UIElement;
```

```
                    }
                    else
                    {
                        returnElement = FindElementRecur-
sive(VisualTreeHelper.GetChild(parent, i) as FrameworkElement, targetType);
                    }
                }
            }
            return returnElement;
        }
        private VisualStateGroup FindVisualState(FrameworkElement element,
string name)
        {
            if (element == null)
                return null;

            IList groups = VisualStateManager.GetVisualStateGroups(element);
            foreach (VisualStateGroup group in groups)
                if (group.Name == name)
                    return group;

            return null;
        }

        private void vgroup_CurrentStateChanging(object sender, Visu-
alStateChangedEventArgs e)
        {
            if (e.NewState.Name == "CompressionTop")
            {

            }

            if (e.NewState.Name == "CompressionBottom")
            {
                //user scrolled to the bottom of the list, let's load addition-
al data.
                LoadAdditionalData();
            }
        }

        //Listbox Scrolling end of list detection
        //http://blogs.msdn.com/b/slmperf/archive/2011/06/30/windows-phone-
mango-change-listbox-how-to-detect-compression-end-of-scroll-states.aspx
        //END OF MSDN CODE

        #endregion

    }

    public class WallPostFriendComparer : IEqualityComparer<WallPost>
    {
        public bool Equals(WallPost x, WallPost y)
        {
            if (x.From.Id == y.From.Id)
            {
                return true;
            }

            else return false;
        }

        public int GetHashCode(WallPost obj)
        {
            return 0;
        }
    }
```

```
    public class WallPostCommentFriendComparer : IEqualityCompar-
er<WallPost.Comment>
    {
        public bool Equals(WallPost.Comment x, WallPost.Comment y)
        {
            if (x.From.Id == y.From.Id)
            {
                return true;
            }

            else return false;
        }

        public int GetHashCode(WallPost.Comment obj)
        {
            return 0;
        }
    }
}
```

D.2.27 Cryptbook.WallUserControl.xaml

```
<UserControl x:Class="Cryptbook.WallUserControl"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="480" d:DesignWidth="480">


    <UserControl.Resources>
        <Style TargetType="ScrollViewer">
            <Setter Property="VerticalScrollBarVisibility" Value="Auto"/>
            <Setter Property="HorizontalScrollBarVisibility" Value="Auto"/>
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="Padding" Value="0"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="BorderBrush" Value="Transparent"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="ScrollViewer">
                        <Border BorderBrush="{TemplateBinding BorderBrush}"
BorderThickness="{TemplateBinding BorderThickness}" Back-
ground="{TemplateBinding Background}">
                            <VisualStateManager.VisualStateGroups>
                                <VisualStateGroup x:Name="VerticalCompression">
                                    <VisualState
x:Name="NoVerticalCompression"/>
                                    <VisualState x:Name="CompressionTop"/>
                                    <VisualState x:Name="CompressionBottom"/>
                                </VisualStateGroup>
                            </VisualStateManager.VisualStateGroups>
                            <Grid Margin="{TemplateBinding Padding}">
                                <ScrollContentPresenter
x:Name="ScrollContentPresenter" Content="{TemplateBinding Content}" ContentTem-
plate="{TemplateBinding ContentTemplate}"/>
```

```
                        </Grid>
                    </Border>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
</UserControl.Resources>


    <Grid x:Name="LayoutRoot">
    <StackPanel Margin="12,0,12,0" x:Name="AddPost" Visibility="Visible" >
        <ListBox x:Name="WallPostListBox" Margin="0,0,0,0" Items-
Source="{Binding}" Visibility="Visible" Height="517" Width="440" Dou-
bleTap="WallPostListBox_DoubleTap">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel>
                        <StackPanel Orientation="Horizontal">
                            <Image Source="{Binding From.Picture}"
Height="40" Width="40" HorizontalAlignment="Left" Margin="0,0,0,0" />

                            <StackPanel Orientation="Vertical">
                                <TextBlock Text="{Binding From.Name}"
x:Name="PostersName" HorizontalAlignment="Left"  TextWrapping="Wrap" Mar-
gin="10,4,0,0" Style="{StaticResource PhoneTextNormalStyle}"/>
                                <StackPanel Orientation="Horizontal"
Width="280">
                                    <TextBlock Text="{Binding GetFriend-
lyCreatedTime}"  x:Name="CreatedTime" HorizontalAlignment="Left"  TextWrap-
ping="Wrap" Margin="10,-4,0,0" Style="{StaticResource PhoneTextSmallStyle}"/>
                                    <TextBlock Text="{Binding GetIsEncrypt-
ed}"  x:Name="IsMessageEncrypted" HorizontalAlignment="Left" FontStyle="Italic"
TextWrapping="Wrap" Margin="10,-4,0,0" Style="{StaticResource PhoneTextSmall-
Style}"/>
                                </StackPanel>
                            </StackPanel>

                            <Button Content="+" HorizontalAlignment="Left"
VerticalAlignment="Center" Width="80" x:Name="CommentButton" Margin="5,0,-10,0"
FontSize="22" Click="CommentButton_Click"></Button>
                            <TextBlock Text="{Binding From.Id}"
x:Name="PostersID"  Visibility="Collapsed" TextWrapping="Wrap" Margin="12,-
6,12,0"   />
                            <TextBlock Text="{Binding Id}"
x:Name="PostID" Visibility="Collapsed" TextWrapping="Wrap" Margin="12,-6,12,0"
/>

                        </StackPanel>

                        <StackPanel Orientation="Vertical">
                            <TextBlock Text="{Binding Message}"  TextWrap-
ping="Wrap" Margin="12,-6,12,0"  Style="{StaticResource PhoneTextNor-
malStyle}"/>
                            <TextBlock Text="{Binding Name}" TextWrap-
ping="Wrap" Margin="12,-6,12,0"  Style="{StaticResource PhoneTextNor-
malStyle}"/>
                            <TextBlock Text="{Binding Description}"
TextWrapping="Wrap" Margin="12,-6,12,0" Style="{StaticResource PhoneTextNor-
malStyle}"/>
                           <HyperlinkButton x:Name="ViewCommentsButton"
HorizontalAlignment="Left" Content="{Binding CommentsList.GetCountForDisplay}"
Click="ViewCommentsButton_Click" FontFamily="Calibri" FontSize="18"/>
                        </StackPanel>

                        <StackPanel>
```

```
                                <ListBox ScrollView-
er.VerticalScrollBarVisibility="Disabled" x:Name="CommentListBox"  Mar-
gin="60,4,0,0" ItemsSource="{Binding CommentsList.CommentsData}" Visibil-
ity="Collapsed" Width="420">
                                <ListBox.ItemTemplate>
                                    <DataTemplate>
                                        <StackPanel>
                                                <StackPanel Orienta-
tion="Horizontal" >
                                                <!-- <Image Source="{Binding
From.Picture}" Height="40" Width="40" HorizontalAlignment="Left" Mar-
gin="0,0,0,0" /> -->

                                                <StackPanel Orienta-
tion="Vertical">
                                                    <TextBlock Text="{Binding
From.Name}"  x:Name="CommentPostersName" HorizontalAlignment="Left" TextWrap-
ping="Wrap" Margin="10,2,0,0" Style="{StaticResource PhoneTextNormalStyle}"/>
                                                        <StackPanel Orienta-
tion="Horizontal"  Width="280">
                                                    <TextBlock
Text="{Binding GetFriendlyCreatedTime}"  x:Name="CommentCreatedTime" TextWrap-
ping="Wrap" Margin="10,-4,0,0" Style="{StaticResource PhoneTextSmallStyle}"/>
                                                    <TextBlock
Text="{Binding GetIsEncrypted}"  x:Name="IsMessageEncrypted"
FontStyle="Italic" TextWrapping="Wrap" Margin="10,-4,0,0"
Style="{StaticResource PhoneTextSmallStyle}"/>
                                                    </StackPanel>
                                                    <TextBlock Text="{Binding
From.Id}" x:Name="CommentPosterID"  Visibility="Collapsed" TextWrapping="Wrap"
Margin="12,-2,12,0"   />
                                                    <TextBlock Text="{Binding
Id}"      x:Name="CommentID" Visibility="Collapsed" TextWrapping="Wrap" Mar-
gin="12,-2,12,0" />
                                                </StackPanel>
                                            </StackPanel>

                                            <TextBlock Text="{Binding Message}"
TextWrapping="Wrap" Margin="10,-6,12,0"  Style="{StaticResource PhoneTextNor-
malStyle}"/>
                                            <TextBlock Text="{Binding Name}"
TextWrapping="Wrap" Margin="10,-6,12,0"  Style="{StaticResource PhoneTextNor-
malStyle}"/>
                                            <TextBlock Text="{Binding Descrip-
tion}" TextWrapping="Wrap" Margin="10,-6,12,0" Style="{StaticResource Pho-
neTextNormalStyle}"/>
                                            <TextBlock  Margin="0,0,0,20"
Text=""></TextBlock>
                                        </StackPanel>
                                    </DataTemplate>
                                </ListBox.ItemTemplate>
                            </ListBox>
                            </StackPanel>
                            <TextBlock  Margin="0,0,0,20" Text=""></TextBlock>
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
            <toolkit:PerformanceProgressBar x:Name="PleaseWaitProgressBar"
IsIndeterminate="True" Visibility="Collapsed"></toolkit:PerformanceProgressBar>
        </StackPanel>
    </Grid>

</UserControl>
```