# 

24.9.2013

**Machbarkeitsstudie**

**S/Chat**

**Verschlüsselter Android Chat**

**Elias Frantar, Wolfram Soyka, Gary Ye**

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Name* | *Datum* | *Unterschrift* |
| *erstellt* | *Elias Frantar* | *17.9.2013* |  |
| *geprüft* |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Version* | *Name* | *QS* | *Datum* | *Kommentar* |
| *0.1* | *Frantar* | *Ye* | *17.9.2013* | *Einführung, Ist-Zustand(Einleitung),Meilensteinplan* |
| *0.2* | *Frantar* | *Ye* | *18.9.2013* | *Projektorganisation, Technologien, Glossar* |
| *0.3* | *Frantar* |  | *21.9.2013* | *Nutzwertanalyse, Soll-Zustand* |
| *0.4* | *Ye* | *Fr.* | *21.9.2013* | *PSP* |
| *0.5* | *Ye* |  | *22.9.2013* | *Produktfunktionen* |
| *0.7* | *Soyka* |  | *22.9.2013* | *Ist Zustand, Persönliche Machbarkeit* |
| *0.8* | *Ye* |  | *24.9.2013* | *Kleine Änderungen, Formatierung* |
|  |  |  |  |  |

# 



Contents

[0](#_Toc367829478)

[1. Einleitung 3](#_Toc367829479)

[2. Ist-Zustand 3](#_Toc367829480)

[2.1. Marktanalyse 3](#_Toc367829481)

[2.2. Trendanalyse 3](#_Toc367829482)

[3. Soll-Zustand 3](#_Toc367829483)

[4. Produktfunktion 4](#_Toc367829484)

[4.1. Verschlüsselung 4](#_Toc367829485)

[4.2. Userfunktionen 8](#_Toc367829486)

[5. Machbarkeit 14](#_Toc367829487)

[5.1. Technische Machbarkeit 14](#_Toc367829488)

[5.1.1. Technologien 14](#_Toc367829489)

[5.1.1.1. Verschlüsselungsalgorithmen 14](#_Toc367829490)

[5.1.1.2. Schlüsselaustauschverfahren 14](#_Toc367829491)

[5.1.1.3. Symmetrisches Verschlüsselungsverfahren 15](#_Toc367829492)

[5.1.1.4. Digitale Signatur 15](#_Toc367829493)

[5.1.1.5. Hashfunktionen 15](#_Toc367829494)

[5.1.2. Umsetzung 15](#_Toc367829495)

[5.2. Wirtschaftliche Machbarkeit 16](#_Toc367829496)

[5.2.1. Personalaufwand 16](#_Toc367829497)

[5.2.2. Investitionsaufwand 16](#_Toc367829498)

[5.2.3. Nutzen 16](#_Toc367829499)

[5.2.4. Risikoanalyse 16](#_Toc367829500)

[5.3. Persönliche Machbarkeit 17](#_Toc367829501)

[5.4. Nutzwertanalyse 18](#_Toc367829502)

[5.4.1. Schlüsselaustauschverfahren 18](#_Toc367829503)

[5.4.2. Symmetrisches Verschlüsselungsverfahren 18](#_Toc367829504)

[5.4.3. Hashfunktion 18](#_Toc367829505)

[6. Projektorganisation 18](#_Toc367829506)

[7. Projektplanung 19](#_Toc367829507)

[a. Projektstrukturplan 19](#_Toc367829508)

[b. Meilensteinplan 20](#_Toc367829509)

[8. Management Summary 20](#_Toc367829510)

[9. Glossar 21](#_Toc367829511)

1. Einleitung

Mit Aufdeckung des NSA-Skandals wird das Thema Datensicherheit zu einem essenziellen Bestandteil der Softwareentwicklung. Aktuell existieren nur sehr wenige Programme, die eine sichere Kommunikation ermöglichen. Von diesen wenigen sind wiederrum nur sehr wenige benutzerfreundlich und werden daher kaum von Privatpersonen eingesetzt.

Deshalb wollen wir in unserem Projekt eine benutzerfreundliche App entwickeln, die es den Usern ermöglicht, sicher Nachrichten zu versenden, ohne jegliches Grundwissen über Verschlüsselungen besitzen zu müssen.

1. Ist-Zustand

Aktuell existieren nur sehr wenige bis gar keine Apps, die eine sichere Kommunikation ermöglichen. Die, die sich als sicher ausgeben haben entweder Sicherheitslücken, oder sind nicht OpenSource und daher nicht Vertrauenswürdig.

* 1. Marktanalyse

Derzeit gibt es zwar bereits mehrere Chat-Systeme (WhatsApp, Viper, iMessage, Blackberry Messenger,...) aber keiner davon ist tatsächlich sicher, bzw. ist es unklar wo, wann und für wie lang gesendete Nachrichten (von dritten) gespeichert werden. Daher wollen wir ein neues, sicheres Chat System entwickeln.

* 1. Trendanalyse

Durch aufdecken des NSA – PRISM Skandals ist vielen Menschen weltweit klar geworden, dass es ihnen die Idee der totalen Überwachung ihrer Kommunikation durch unkontrollierbare Behörden von verschiedenen Regierungen.

Durch diesen Aufschrei der Weltbevölkerung ist klar geworden, dass der Trend in Zukunft klar in Richtung verschlüsselter Datenübertragung geht.

1. Soll-Zustand

**Muss – Ziele:**

1. **Sicherheit**

Die Sicherheit der Nachrichten ist der wichtigste Punkt dieses Projektes, deshalb ist die Implementierung geeigneter Verschlüsselungssysteme (internationale Standards: RSA, AES, SHA3) und Protokolle essenziell.

1. **Bedienbarkeit**

Die App soll benutzerfreundlich und damit leicht bedienbar sein. Wichtig hierbei ist, dass ein User ohne Kenntnisse über Verschlüsselung die App problemlos und ohne Zusatzaufwand bedienen kann.

1. **Zuverlässigkeit**

Das Verschicken der Nachrichten soll zuverlässig funktionieren. D.h. konkret, dass bei guter Internetverbindung rund 95% aller Nachrichten beim ersten Versenden ankommen und 99% beim (automatischen) Wiederholen.

**Kann – Ziele:**

1. **Kompatibilität**

Die App soll von Android-Version 4.x bis 2.3 abwärtskompatibel sein ohne etwaige Funktionalitäten zu verlieren.

1. **Gruppen**

Die App soll nicht nur Nachrichten mit einer Person, sondern auch in Gruppen von mehreren Personen gleichzeitig austauschen können.

**Nicht – Ziele**:

1. **Design**

Ein spezielles modernes Design der Applikation für das extra angefertigte Bedienungselemente notwendig sind.

1. **Funktionen**

Bereitstellen spezielle Zusatzfunktionen, wie das Versenden von Bild oder Videodateien oder das Erstellen eines detaillierten persönlichen Kontaktprofils

1. Produktfunktion
   1. Verschlüsselung

Sämtliche Nachrichten sollen verschlüsselt versendet und abgespeichert werden. Zusätzlich werden sie noch digital signiert. Dazu müssen zahlreiche verschiedene Schlüssel („Sessionkeys“) generiert werden.

/**LF010**/ **Private / Public Key-pair generieren**

Beim erstmaligen Starten der Applikation wird ein zufälliges Schlüsselpaar generiert. Der öffentliche Teil des Schlüsselpaares wird mit Kommunikationspartnern ausgetauscht. Der private Teil wird zur Entschlüsselung sämtlicher Nachrichten verwendet.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Private / Public-Key-pair generieren (LF010)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Beim erstmaligen Starten der Applikation wird ein zufälliges Schlüsselpaar generiert. Der öffentliche Teil des Schlüsselpaares wird mit Kommunikationspartnern ausgetauscht. Der private Teil wird zur Entschlüsselung sämtlicher Nachrichten verwendet.  *Auslöser:* die Applikation wird zum ersten Mal gestartet  *Ergebnis:* der Benutzer erhält einen persönlichen öffentlichen und privaten Schlüssel  *Akteure:* das neue System  *Eingehende Informationen:* ein Zufallswert aus Zeit und zufälligen Speicherdaten  *Vorbedingungen:* es ist noch kein Key generiert worden  *Nachbedingungen:* der Schlüssel kann nun mit anderen Personen ausgetauscht werden | hoch | mittel | Must Have |

/**LF020**/ **Sessionkeys generieren**

Zum Verschlüsseln jeder Nachricht muss ein eigener symmetrischer Sessionkey zufällig generiert werden.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Sessionkeys generieren (LF020)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Zum Verschlüsseln jeder Nachricht muss ein eigener symmetrischer Sessionkey zufällig generiert werden.  *Auslöser:* Es soll einen neue Nachricht verschlüsselt werden  *Ergebnis:* die Nachricht wird mit diesem Key verschlüsselt  *Akteure:* das System  *Eingehende Informationen:* der Masterkey  *Vorbedingungen:* es muss bereits ein Masterkey generiert worden sein  *Nachbedingungen:* die Nachricht kann mit diesem Schlüssel verschlüsselt werden | hoch | mittel | Must Have |

/**LF030**/ **Nachrichten signieren**

Jede zu versendende Nachricht muss digital signiert werden, damit der Empfänger ihren Absender eindeutig feststellen kann.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Nachricht signieren (LF030)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Jede zu versendende Nachricht muss digital signiert werden, damit der Empfänger ihren Absender eindeutig feststellen kann.  *Auslöser:* es soll eine Nachricht versendet werden  *Ergebnis:* die zu versendende Nachricht ist digital signiert  *Akteure:* das System  *Eingehende Informationen:* der eigene private-Key  *Vorbedingungen:* es muss bereits ein private-Key generiert worden sein  *Nachbedingungen:* die Nachricht ist nun digital signiert | hoch | mittel | Must Have |

## Userfunktionen

Der User kann folgende Funktionen der App benutzen.

/**LF110**/ **Neuen Schlüssel erstellen**

Ein User kann auf Wunsch ein neues Schlüsselpaar generieren lassen. Es ist zu beachten, dass alle alten Konversationen nach Benutzung dieser Funktion erst nach erneutem Schlüsselaustausch wieder fortgeführt werden können. (wollten wir nicht automatisch dessen Kontakte informieren?)

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Neuen Schlüssel erstellen (LF110)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Ein User kann auf Wunsch ein neues Schlüsselpaar generieren lassen. Es ist zu beachten, dass alle alten Konversationen nach Benutzung dieser Funktion erst nach erneutem Schlüsselaustausch wieder fortgeführt werden können.  *Auslöser:* der User wünscht sich ein neues Schlüsselpaar.  *Ergebnis:* der User erhält ein neues, persönliches Schlüsselpaar  *Akteure:* der User  *Eingehende Informationen:* keine  *Vorbedingungen:* es muss bereits einmal ein Key generiert worden sein  *Nachbedingungen:* der User hat einen neuen Key, den er erneut an seine Kontakte weiterleiten muss | mittel | mittel | Nice to Have |

/**LF120**/ **Nachrichten versenden**

Ein User kann eine Nachricht verschlüsselt an einen Kontakt verschicken. Eine Nachricht besitzt eine maximale Länge.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Nachrichten versenden (LF120)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Ein User kann eine Nachricht verschlüsselt an einen Kontakt verschicken. Eine Nachricht besitzt eine maximale Länge.  *Auslöser:* der User möchte eine Nachricht verschicken  *Ergebnis:* der Empfänger bekommt eine Nachricht  *Akteure:* der Sender und der Empfänger  *Eingehende Informationen:* die Nachricht und der Empfänger  *Vorbedingungen:* die Nachricht überschreitet nicht die vorgegebene maximale Länge, es muss bereits ein Schlüssel generiert worden sein  *Nachbedingungen:* der Empfänger erhält die Nachricht | hoch | hoch | Must Have |

/**LF130**/ **Nachrichten anzeigen**

Beim Öffnen eines Chats mit einem anderen User werden alle alten abgespeicherten Nachrichten automatisch geladen und angezeigt.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Nachrichten anzeigen (LF130)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Beim Öffnen eines Chats mit einem anderen User werden alle alten abgespeicherten Nachrichten automatisch geladen und angezeigt  *Auslöser:* der User möchte die alte Nachrichten lesen  *Ergebnis:* dem User werden die alten Nachrichten angezeigt  *Akteure:* der User  *Eingehende Informationen:* der Kontakt, dessen Nachrichten angezeigt werden sollen  *Vorbedingungen:* es müssen bereits Nachrichten ausgetauscht worden sein  *Nachbedingungen:* der User kann die Nachrichten nun lesen | hoch | gering | Must Have |

/**LF140**/ **Nachrichten löschen**

Ein User kann eine seiner gespeicherten Nachrichten löschen. Die Nachricht wird natürlich nur auf seinem Gerät gelöscht nicht aber auf dem des Kommunikationspartners.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Nachrichten löschen (LF140)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Ein User kann eine seiner gespeicherten Nachrichten löschen. Die Nachricht wird natürlich nur auf seinem Gerät gelöscht nicht aber auf dem des Kommunikationspartners.  *Auslöser:* der User möchte eine Nachricht löschen  *Ergebnis:* die Nachricht ist lokal gelöscht  *Akteure:* der User  *Eingehende Informationen:* die zu löschende Nachricht  *Vorbedingungen:* es muss eine Nachricht vorhanden  *Nachbedingungen:* die Nachricht kann nicht mehr angezeigt werden | gering | gering | Must Have |

/**LF150**/ **Kontakte hinzufügen**

Ein User kann einen neuen Kontakt durch Eingabe dessen Publickeys seiner Kontaktliste hinzufügen.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Kontakte hinzufügen (LF150)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Ein User kann einen neuen Kontakt durch Eingabe dessen Publickeys seiner Kontaktliste hinzufügen.  *Auslöser:* der User möchte einen neuen Kontakt hinzufügen  *Ergebnis:* der neue Kontakt ist in der Kontaktliste hinzugefügt  *Akteure:* der User  *Eingehende Informationen:* der neue Kontakt  *Vorbedingungen:* der neue Kontakt darf nicht in der Kontaktlisten vorhanden sein  *Nachbedingungen:* es können nun Nachrichten mit diesem Kontakt ausgetauscht werden | hoch | gering | Must Have |

/**LF140**/ **Kontakte löschen**

Ein User kann einen Kontakt aus seiner Kontaktliste löschen. Dadurch werden auch sämtliche mit diesem Kontakt ausgetauschte Nachrichten gelöscht.

|  |  |  |  |
| --- | --- | --- | --- |
| Funktion | Nutzen | Aufwand | Priorität |
| UseCase:  *Name:* Kontakte löschen (LF140)  *Art:* Anwendungsfall  *Kurzbeschreibung:* Ein User kann einen Kontakt aus seiner Kontaktliste löschen. Dadurch werden auch sämtliche mit diesem Kontakt ausgetauschte Nachrichten gelöscht.  *Auslöser:* der User möchte einen Kontakt von der Kontaktliste entfernen  *Ergebnis:* der zu löschende Kontakt wird von der Kontaktliste entfernt  *Akteure:* der User  *Eingehende Informationen:* der zu löschende Kontakt  *Vorbedingungen:* der zu löschende Kontakt muss auf der Kontaktliste des Benutzers vorhanden sein  *Nachbedingungen:* der Kontakt ist nicht mehr auf der Kontaktliste vorhanden und es sind keine mit ihm ausgetauschten Nachrichten mehr abgespeichert | gering | gering | Must Have |

1. Machbarkeit
   1. Technische Machbarkeit
      1. Technologien
         1. Verschlüsselungsalgorithmen

Für dieses Projekt werden kryptografische Algorithmen verschiedenster Art verwendet werden. Konkret werden ein Verfahren für den Schlüsselaustausch, ein symmetrisches Verschlüsselungsverfahren, ein digitales Signaturverfahren und eine kryptografische Hashfunktion benötigt.  
Alle im folgenden Abschnitt behandelte Algorithmen gelten als sicher, d.h. sie werden schon über ein längeren Zeitraum verwendet, von Experten analysiert und es wurden bisher noch keine Sicherheitslücken entdeckt.   
  
(Alle im folgenden Abschnitt behandelten kryptografische Begriffe können im Glossar gefunden werden)

* + - 1. Schlüsselaustauschverfahren

Grundsätzlich gibt es für den sicheren Schlüsselaustausch zwei verschiedene Möglichkeiten: ein asymmetrisches Verschlüsselungsverfahren oder der Diffie-Hellman-Schlüsselaustausch.  
Letzterer hat den entscheidenden Nachteil, dass mit jedem einzelnen Kommunikationspartner ein eigener gemeinsamer Schlüssel generiert werden muss. Außerdem kann ein asymmetrisches Verfahren gleichzeitig zur Verschlüsselung, Signierung und Authentifizierung verwendet werden. Deshalb wird ein asymmetrisches Verfahren verwendet werden.

Hier gibt es eigentlich auch nur zwei infrage kommende Möglichkeiten: der bewehrte und standardisierte RSA oder ein Verfahren auf elliptischen Kurven basierendes Verfahren (ECC).  
Erstere wird schon seit den 90iger-Jahren von zahlreichen Kryptoprodukten eingesetzt und hat bisher noch keine (bei fehlerfreier Implementierung) sicherheitskritische Schwäche offenbart. Ein Nachteil dieses Verfahrens ist aber, dass sehr lange Schlüssel benötigt werden (2048bit) um eine hohe Sicherheit zu garantieren. Des Weiteren ist die Schlüsselgenerierung keinesfalls trivial, da hierzu sehr lange voneinander unabhängige zufällige Primzahlen gefunden werden müssen. Im Gegensatz dazu hat ein ECC-Verfahren den entscheidenden Vorteil, dass es vergleichbare Sicherheit mit wesentlich geringerer Schlüssellänge und damit wesentlich höherer Performance ermöglicht. Die Kryptografie auf Basis elliptischer Kurven ist aber relativ neu und wurde bisher kaum eingesetzt und ist daher vergleichsweise weniger analysiert und damit weniger bewehrt als RSA. Außerdem gibt es in diesem Bereich noch kaum international anerkannte Standards. Aus diesem Grund werden wir in unserer Applikation das RSA-Verfahren einsetzen, um ein höheres Vertrauen der User in unsere Applikation zu erlangen.

* + - 1. Symmetrisches Verschlüsselungsverfahren

Grundsätzlich gibt es Blockchiffren und Stromchiffren, wobei erste viel bewehrter sind und ein Schlüssel (unter richtiger Anwendung) ohne Sicherheitsrisiko mehrmals verwendet werden kann. Deshalb werden wir eindeutig ein symmetrisches Verfahren implementieren.

Hier gibt es sehr viele Möglichkeiten, wobei die bekanntesten und damit bewährtesten wohl die Finalisten der AES (Advanced Encryption Standard)-Wettbewerbs sind. Da der Sieger dieses Wettbewerbs seit 2001 in zahlreichen Kryptoprodukten implementiert wurden und sich auch nach dem Wettbewerb trotz zahlreicher Analyseversuche keine Sicherheitslücken gefunden wurden, werden wir auch auf diesen Algorithmus (AES) setzen, da dieser wahrscheinlich auch bei den Usern auf die höchste Akzeptanz stößt.

* + - 1. Digitale Signatur

Da der sowieso verwendet RSA auch als bewehrtes Signaturverfahren eingesetzt werden kann und die beste Alternative der DSA (Digital Signature Standard) einen weiteren Schlüssel pro Kommunikationspartner erfordert, werden wir auch hier RSA verwenden.

* + - 1. Hashfunktionen

Für das Projekt wird zuerst einmal eine schlüsselabhängige Hashfunktion verwendet. Grundsätzlich kann jede Blockchiffre (in unserem Fall AES) in einer speziellen Betriebsart als schlüsselabhängige Hashfunktion eingesetzt werden. Deshalb werden wir auch hier den AES einsetzen, um auf zusätzliche redundante Algorithmen verzichten zu können.

Für eine schlüsselunabhängige Hashfunktion gibt es viele verschiede gute Algorithmen, wie z.B. SHA2, SHA3, SHA-512, die sich nicht wesentlich voneinander unterscheiden. Daher kann das Verfahren nach persönlicher Präferenz ausgewählt werden. Wir haben uns für den neuesten, 2012 gewählten Sieger der SHA(Secure Hash Algorithm) Challenge, Keccak bzw. SHA3, da wir nur standardisierte und von vielen Experten geprüfte Algorithmen verwenden wollen.

Die genauen Entscheidungskriterien für die einzelnen Verschlüsselungsalgorithmen sind der Nutzwertanalyse zu entnehmen.

* + 1. Umsetzung

Beim Nachrichtenaustausch zwischen zwei verschiedenen Systemen, müssen Server-Client- Verbindungen hergestellt werden. Diese können konstruiert werden, indem sich Sockets auf Basis des Internet Protocol’s (IP) miteinander verbinden. Die Socket-Adresse des Servers ist statisch und verändert sich in den seltentesten Fällen - hingegen sind die Socket-Adressen der Clients dynamisch. Die Server-Client Verbindung ist die sinnvollste Entscheidung, da der Versender keine Information über die, sich im Laufe der Zeit ändernde Adresse des Empfängers verfügen muss, um Nachrichten an ihn zu verschicken. Stattdessen muss er nur die des Servers kennen, denn der Server verwaltet eine Liste von den aktuellsten Adressen seiner Clients.

* 1. Wirtschaftliche Machbarkeit
     1. Personalaufwand

Der Umfang der Software ist relativ groß, denn es müssen eine Android Application erstellt sowie ein Server aufgesetzt werden. Außerdem muss jede Nachricht mit zahlreichen Verschlüsselungssystemen gesichert werden. Dies wird einiges an Durchführungszeit benötigen.

Die Gesamtstundenanzahl beträgt ungefähr 200 Personenstunden. Die Aufgaben werden um dies einzuhalten und um möglichst effizient zu arbeiten gleichmäßig auf alle drei Projektmitglieder aufgeteilt.

* + 1. Investitionsaufwand

Die Investitionsaufwände sind relativ gering, da der Großteil der verwendeten Softwarepakete opensource und damit kostenlos ist.

Das Projekt wird in Wien in Räumlichkeiten des TGM bzw. bei den Teammitgliedern auf deren Rechnern durchgeführt. Daher fallen hier keine zusätzlichen Kosten an.

* + 1. Nutzen

Das System wird als Android Application unter einer OpenSource License veröffentlicht und ist daher für alle User der Android Plattform (aber Version 2.3) benutzt werden.

* + 1. Risikoanalyse

***Krankheitsbedingter Ausfall:***

Ein Teammitglied kann auf Grund einer Krankheit bestimmte Aufgaben nicht erfüllen.

* Auswirkung: **niedrig**, Eintrittswahrscheinlichkeit: **hoch**
* Präventive Maßnahmen: Um Ausfällen die Auswirkung auf das Projekt zu nehmen, wird die Arbeit von Anfang an auf alle Team Mitglieder gleichmäßig aufgeteilt.
* Reaktive Maßnahmen: Die Aufgaben des ausgefallenen Teammitglieds werden temporär von den anderen übernommen.

***Datenverlust****:*

Die aktuellen Arbeitsdaten gehen verloren und können nicht wiederhergestellt werden.

* Auswirkung: **hoch**, Eintrittswahrscheinlichkeit: **sehr gering**
* Präventive Maßnahmen: Sämtliche Daten werden in der Cloud versioniert und gesichert.
* Reaktive Maßnahmen: Es werden die zuletzt gesicherten Daten wiederhergestellt und weiterverwendet.

***Serverprobleme:***

Ein Server fällt aus und kann nicht sofort repariert werden.

* Auswirkung: **hoch**, Eintrittswahrscheinlichkeit: **sehr gering**
* Präventive Maßnahmen: Da das Ausfallen eines Servers nicht vorhersehbar / verhinderbar ist, können lediglich Ersatzserver bereitgehalten werden.
* Reaktive Maßnahmen: Es wird auf einen Ersatzserver umgestiegen.

***Netzwerkprobleme***:

Das Internet fällt aus, der Server ist unerreichbar oder gesendete Nachrichten kommen nicht beim Empfänger an.

* Auswirkung: **hoch**, Eintrittswahrscheinlichkeit: **gering**
* Präventive Maßnahmen: Netzwerkprobleme sind unvorhersehbar.
* Reaktive Maßnahmen: Abwarten oder ggf. Bugfixing
  1. Persönliche Machbarkeit

Der schwierigste Teil, das Sichern des Systems durch kryptographische Algorithmen, stellt kein Problem dar, denn dafür ist in unserem Team ein absoluter Experte zuständig. Das Wissen über die Webseiten- und Datenbankkreation ist auch vorhanden. Das Team hat zwar nicht sehr viele Erfahrungen in diesen zwei Bereichen, aber die Aufgabe ist nicht zu komplex, daher wird dies kein Hindernis darstellen.

* 1. Nutzwertanalyse
     1. Schlüsselaustauschverfahren

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Schlüsselaustauschverfahren** | | RSA | | Diffie-Hellman | | ECC | |
| Kriterium | % | Rang | G \* R | Rang | G \* R | Rang | G \* R |
| Sicherheit | 20 | 1 | 20 | 1 | 20 | 1 | 20 |
| Performance | 10 | 2 | 20 | 2 | 20 | 1 | 10 |
| Schlüssellänge | 10 | 3 | 30 | 2 | 20 | 1 | 10 |
| Bewährtheit | 40 | 1 | 40 | 2 | 80 | 3 | 120 |
| Schlüsselmanagement | 20 | 1 | 20 | 3 | 60 | 1 | 20 |
| Gesamt | 100 | **1** | 130 | 3 | 200 | 2 | 180 |

* + 1. Symmetrisches Verschlüsselungsverfahren

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Symmetrisches Kryptoverfahren** | | AES | | Twofish | | RC6 | |
| Kriterium | % | Rang | G \* R | Rang | G \* R | Rang | G \* R |
| Sicherheit | 30 | 1 | 30 | 1 | 30 | 1 | 30 |
| Performance | 30 | 1 | 30 | 3 | 90 | 2 | 60 |
| Bewährtheit | 40 | 1 | 40 | 2 | 80 | 2 | 80 |
| Gesamt | 100 | **1** | 70 | 3 | 200 | 2 | 170 |

* + 1. Hashfunktion

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **Hashfunktion** | | SHA-3 | | SHA-512 | | SHA-2 | |
| Kriterium | % | Rang | G \* R | Rang | G \* R | Rang | G \* R |
| Sicherheit | 20 | 1 | 20 | 1 | 20 | 1 | 20 |
| Performance | 30 | 1 | 30 | 3 | 90 | 2 | 60 |
| Userakzeptanz | 25 | 1 | 25 | 2 | 50 | 2 | 50 |
| Bewährtheit | 25 | 3 | 75 | 1 | 25 | 1 | 25 |
| Gesamt | 100 | **1** | 150 | 3 | 185 | 2 | 155 |

1. Projektorganisation

Die Projektauftraggeber und Ansprechpartner dieses Projekts sind Dr. Gottfried Koppensteiner und Erhard List.

**Gary Ye:**

|  |  |
| --- | --- |
| *Position:* | Projektleiter |
| *Spezialgebiet:* | Algorithmen |
| *Aufgabenbereich:* | Programmieren und Konfigurieren der Kommunikation über das Internet, Implementieren der Datenspeicherung, Projektleitung |

**Elias Frantar**:

|  |  |
| --- | --- |
| *Position:* | Teammitglied |
| *Spezialgebiet:* | Kryptographie |
| *Aufgabenbereich:* | Planung und Implementierung der kryptografischen Systeme, Android Entwicklung |

**Wolfram Soyka**:

|  |  |
| --- | --- |
| *Position:* | Teammitglied |
| *Spezialgebiet:* | Android |
| *Aufgabenbereich:* | Android Entwicklung, Unterstützung in allen Aufgabenbereichen |

Gottfried Koppensteiner

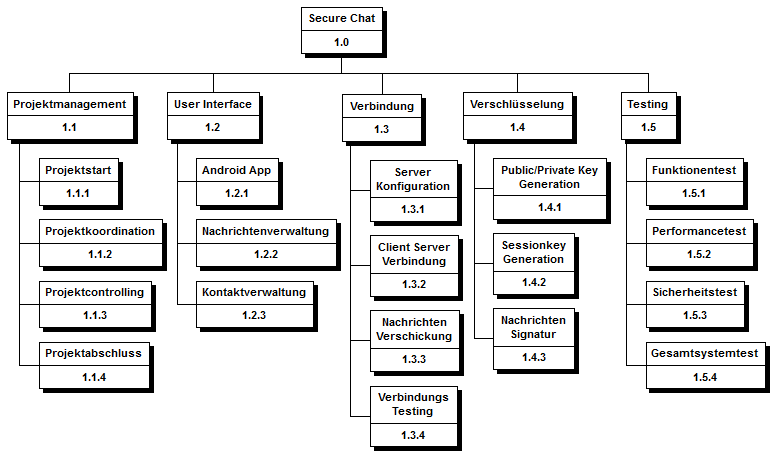
Erhard List

Gary Ye

Elias Frantar

Wolfram Sokya

1. Projektplanung
   1. Projektstrukturplan



* 1. Meilensteinplan

|  |  |  |
| --- | --- | --- |
| **Meilenstein** | **Deliverable** | **Datum** |
| Start | Alle Hefte mit enthaltenen Informationen | 01.09.2013 |
| Android Skelett fertiggestellt | Eine App, die alle grundlegenden GUI-Elemente enthält, aber weder ein gutes Design noch Funktionen bereitstellt | 07.10.2013 |
| Versenden von Nachrichten über das Internet funktioniert | Es können Textnachrichten zwischen zwei Geräten über das Internet (unverschlüsselt versendet werden) | 21.10.2013 |
| Implementierung der Verschlüsselungssysteme abgeschlossen | Die Nachrichten können nun verschlüsselt versendet werden | 15.11.2013 |
| Implementierung der Datenspeicherung abgeschlossen | Die ausgetauschten Nachrichten können nun abgespeichert und wieder geladen werden. | 06.12.2012 |
| Gesamtsystemtest abgeschlossen | Fertige App | 15.12.2013 |

1. Management Summary

Das Projektteam möchte einen Chat entwickeln, der auf Android läuft und sichere Kommunikation in den Alltag bringt.

Es soll also eine Android Application entwickelt werden, die es den Usern erlaubt verschlüsselte Nachrichten über das Internet zu versenden. Es gibt zwar bereits sehr viele ähnliche Applikationen, aber keines entspricht den gewünschten Sicherheitskriterien. Deshalb wird eine speziell abgestimmte App entwickelt.

Zur Übertragung der Nachrichten soll ein Server-Client System eingesetzt werden.

Bei der Verschlüsselung wird auf bewährte Standards (RSA, AES, SHA3) gesetzt.   
Die genauen Entscheidungskriterien sind der Sektion „Technische Machbarkeit“ zu entnehmen.

Die Umsetzung ist sowohl technisch, wirtschaftlich als auch persönlich machbar. Das Team und die Arbeitsmittel sind ausreichend. Etwaige Wissenslücken können sehr schnell und ohne großen Zeitverlust geschlossen werden. Das Know – How ist vor allem im komplexesten und schwierigsten Teil, nämlich der Sicherheit des Systems, ausreichend vorhanden.

Der Server wird durch den Projektleiter (privat) zur Verfügung gestellt, daher fallen keine extra Kosten an.

Das komplette Projekt wird ca. 2,5 Monate dauern, genaue Termine sind in der Meilensteinplanung ersichtlich.

Alles in allem sollte das Projekt durchführbar sein.

1. Glossar

* **Asymmetrische Verschlüsselung:** Ver- und Entschlüsselungen geschehen durch unterschiedliche Schlüssel (bei symmetrischen Verfahren durch denselben Schlüssel)
* **Public/Private Key-pair:** Ein asymmetrisches Verschlüsselungsverfahren benötigt immer einen öffentlichen Schlüssel zur Verschlüsselung und einen privaten Schlüssel zur Entschlüsselung von Nachrichten. Der öffentliche Schlüssel ist allen Sendern bekannt, der private aber nur dem Empfänger.
* **Session Key:** Jede Nachricht wird aus Performancegründen nur mit einem symmetrischen Sessionkey verschlüsselt, der anschließend asymmetrisch verschlüsselt wird.
* **Digitale Signatur:** Mithilfe von asymmetrischen Verschlüsselungsverfahren können Nachrichten digital signiert werden
* **Stromchiffre:** Eine Chiffrenart, die beliebig große Daten verschlüsseln kann. Dabei wird jedes Bit der Daten einzeln verschlüsselt. Ein Schlüssel kann nur einmal verwendet werden
* **Blockchiffre:** Eine Chiffrenart, die einen Datenblock bestimmter Größe verschlüsselt. Ein Schlüssel kann mehrmals verwendet werden.
* **Diffie-Hellman-Schlüsselaustausch:** Ein auf dem diskreten Logarithmus basierendes Verfahren zur geheimen Generierung eines gemeinsamen Schlüssel
* **RSA (Rivest Shamir Adelmann):** Ein asymmetrisches Verschlüsselungsverfahren dessen Sicherheit auf dem Faktorisierungsproblem beruht, mit einem privaten und einem öffentlichen Schlüssel
* **ECC (Elliptic Curve Cryptography):** Ein public-key-Kryptographieverfahren auf Basis elliptischer Kurven
* **AES (oder Rijndael):** Der Sieger des Blockchiffrenstandardisierungswettbewerbs für den *Advanced Encryption Standard*
* **SHA3 (Keccak):** Der Sieger des Einweghashfunktionsstandardisierungswettbewerbs für den *Secure Hash Algorithm 3*
* **SHA2, SHA512:** andere sichere kryptografische Einweghashfunktionen