Entwurf und Implementierung eines Systems für Einmalpasswörter

Facharbeit von Folke Will

zur Erlangung des Abschlusses

Staatlich geprüfter Techniker

im Fachbereich

Informations- und Kommunikationstechnik

Unterrichtsfach:	Entwickeln und Programmieren
Betreuender Fachlehrer:	Dr. Thomas Bogner
Klasse:	FIT7H
Druckdatum:	24. April 2009
eingereicht am:	27. April 2009

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Im Rahmen dieser Facharbeit wird ein System entworfen, das die Authentifizierung von Benutzern mithilfe von Einmalpasswörtern vornehmen kann. Ein Einmalpasswort ist eine spezielle Art eines Passworts, das *nur einmal* und *nur für eine bestimmte Zeit* gültig ist [Tan01, Seiten 599 f.]. Dies ist in sicherheitskritischen Bereichen sinnvoll, in denen das Passwort unter Anwesenheit Fremder eingegeben werden muss, beispielsweise für mobile Mitarbeiter, die in fremden Gebäuden arbeiten und sich am Netzwerk ihrer Firma anmelden wollen.

Der Benutzer verwendet zur Authentisierung mit Einmalpasswörtern also nicht nur etwas, das er *kennt* (ein Passwort), sondern muss zusätzlich im *Besitz* der Sache sein, welche die Einmalpasswörter darstellt. Dadurch wird die Sicherheit erhöht. Wird der Benutzer bei der Eingabe des Einmalpassworts beobachtet, so ist dies für die Sicherheit nicht kritisch, da das benutzte Einmalpasswort ausschließlich für diesen Vorgang gültig war.

Begriffe Der Benutzer, der das System zur Anmeldung nutzt, wird im folgenden als *authentisierender Benutzer* bezeichnet. Die Stelle, die den Benutzer authentifiziert, wird als *authentifizierende Stelle* bezeichnet. Abbildung 1 zeigt die Begriffe im Zusammenspiel. Der Benutzer authentisiert sich mit dem aktuellen Einmalpasswort am System. Das System kennt ebenfalls das aktuelle Einmalpasswort und kann den Benutzer durch einen Vergleich mit dem eingegebenen Einmalpasswort authentifizieren.

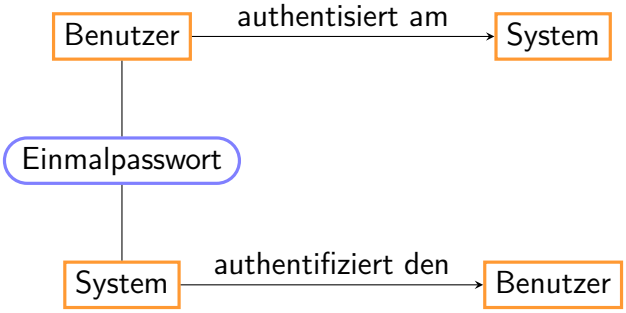


Abbildung 1: Darstellung der Begriffe authentifizieren und authentisieren

1.1 Problemstellung

Mobile Mitarbeiter, die mit einem Laptop bei Kunden arbeiten, melden sich von dort häufig am Intranet ihrer Firma an. Dabei können sie einerseits aktiv von Fremden beobachtet werden, andererseits aber auch passiv, beispielsweise durch Überwachungskameras. Wird der Mitarbeiter bei der Eingabe seines Passworts beobachtet, fällt es einem möglichen Angreifer leicht, einen unbefugten Zugriff zu einem System der Firma des Mitarbeiters zu

erhalten, da die Kombination aus Benutzername und Passwort häufig ausreicht, um sich an einem System zu authentisieren.

Auch private Anwender sind zunehmend von diesem Problem betroffen. Der Bundestag hat im November 2008 ein Gesetz verabschiedet, das es dem Bundeskriminalamt gestattet, heimlich auf private Computer zuzugreifen und dort Software zu installieren [Kre08a]. Diese Software besitzt unter anderem Keylogger-Funktionalitäten, die

> *. . . unter anderem die Tastatureingaben für Passwörter, Login-Daten und PINs vor einer möglichen Verschlüsselung von Informationen aufzeichnen. Davon erhofft sich das BKA, alle Zugangsdaten für genutzte Dienste per Fernübertragung frei Haus geliefert zu bekommen.* [Kre08b]

1.1.1 Abgrenzung des Themas

Diese Facharbeit stellt einen theoretischen Ansatz der Authentifizierung mit Einmalpasswörtern sowie Teile der praktischen Umsetzung dar. Es werden keine Programme entwickelt, die Einmalpasswörter verwenden, sondern vielmehr eine allgemeine Schnittstelle für verschiedene Systeme, über welche die Einmalpasswörter implementiert werden können. Dazu werden einige Beispiele entwickelt und demonstriert.

1.2 Relevanz des Themas

Von dem Problem der unsicheren Anmeldung ist jede Firma mit mobilen Mitarbeitern sowie jede Privatperson, deren Computer über einen Zugang zum Internet verfügt, betroffen. Das Fachmagzin Heise-Online berichtete bereits 2005, dass die Gefahr durch Keylogger unterschätzt werde und weiter zunehme [Bru05].

Firmen sind durch Wirtschaftsspionage auch in den eigenen Gebäuden Angriffen ausgeliefert. So wäre es zum Beispiel möglich, dass eine Reinigungskraft heimlich einen Hardware-Keylogger an einem System anbringt und am nächsten Tag ausliest. Ein Hardware-Keylogger ist ein sehr kleines Gerät, häufig in Form eines Steckers, das zwischen Tastatur und Computer angebracht wird und alle Eingaben der Tastatur speichert.

Die Mindmap in Abbildung 2 auf der nächsten Seite zeigt einige der möglichen Sicherheitsrisiken bei der Verwendung konventioneller Passwörter.

Alle dort genannten Risiken können durch die Verwendung von Einmalpasswörtern vermieden werden, da die Bedeutung eines einzelnen Einmalpassworts sofort nach dessen Verwendung erlischt.

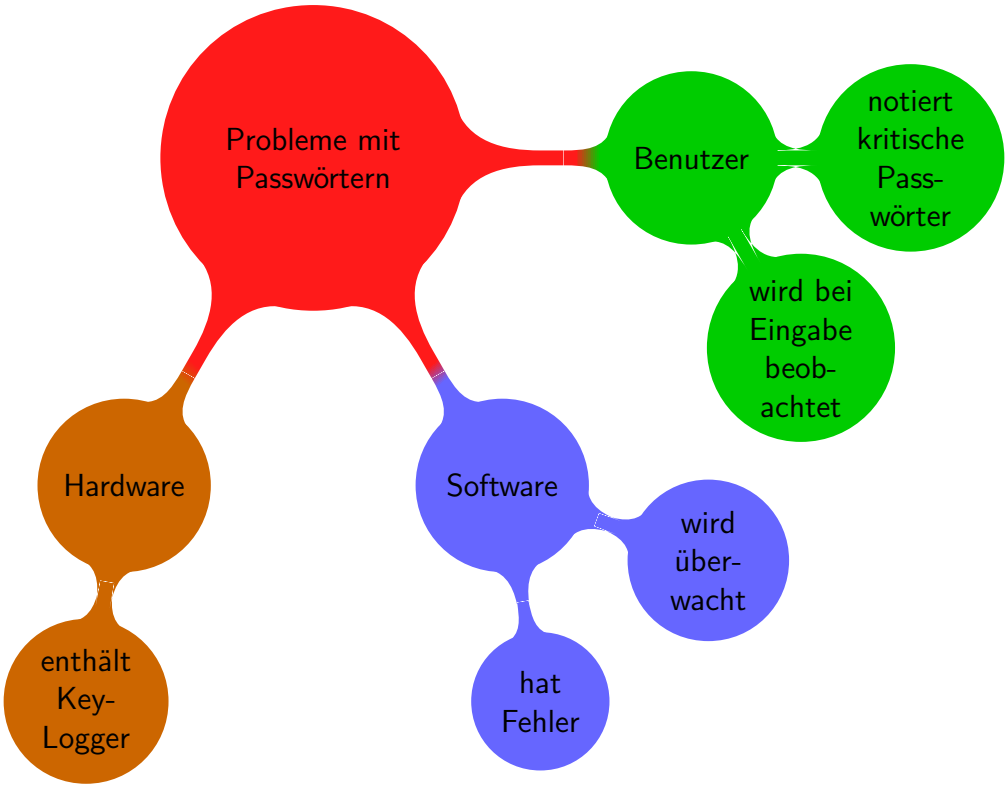


Abbildung 2: Mögliche Sicherheitsrisiken bei konventionellen Passwörtern

1.3 Stand des Problems in der Fachliteratur

Die Problematik der Unsicherheit von konventionellen Passwörtern ist in der Fachliteratur bekannt, beispielsweise in [Tan01, Seite 592] oder [Sch96, Seite 62]. Zusammenfassend wird berichtet, dass Passwörter unsicher sind und durch etwas ergänzt werden sollten, das der Benutzer *besitzt*, beispielsweise eine TAN-Liste, oder etwas, das der Benutzer *ist*, wie beispielsweise seinen Fingerabdruck.

Da Einmalpasswörter von einem Gerät oder einer Liste abgelesen werden, stellen sie etwas dar, das der Benutzer besitzt. Es gibt bereits verschiedene Methoden zur Authentifizierung mit Einmalpasswörtern.

1.3.1 SKEY

In [Sch96, Seite 64] wird ein System names *SKEY* beschrieben, bei dem jeder Benutzer eine Liste mit 100 Einmalpasswörtern erhält, die er nacheinander für seine Anmeldungen benutzt. Der hohe Verwaltungsaufwand ist der Nachteil dieses Systems: jeder Benutzer muss nach 100 Anmeldungen eine neue Liste erhalten, welche die nächsten 100 Einmalpasswörter enthält. Banken verwenden ähnliche Verfahren häufig für Online-Banking (TAN-Listen).

1.3.2 RFC2289

In [HMNS98] wird das Konzept eines ähnliches Systems beschrieben, das auf gedruckte Listen verzichtet. Stattdessen verwendet es ein Challenge-Response-Verfahren, bei dem die authentifizierende Stelle dem Benutzer eine Zeichenkette mitteilt (Challenge), für die ein Programm auf seiner Seite zusammen mit einer geheimen Zeichenkette (Secret) das Einmalpasswort berechnet (Response). Der Aufwand ist also hoch, zusätzlich besteht auch hier der Nachteil, dass das System nach einer konstanten Anzahl von Anmeldungen neu initialisiert werden muss.

1.3.3 RSA SecurID

Weit verbreitet ist das System SecurID von RSA Security Inc., dem Entwickler des RSA-Kryptosystems. Bei diesem System wird jeder Nutzer mit einem sogenannten Token ausgestattet. Dies ist ein kleines Gerät, welches das aktuell gültige Passwort anzeigt, das sich jede Minute ändert. Ein Nachteil dieser Lösung sind die hohen Kosten. Jedes Token muss bezahlt werden, zusätzliche entstehen Lizenzkosten von mehreren Tausend US-Dollars. Nähere Informationen zu diesem System finden sich in [RS09].

2 Konzept

Basierend auf der Problemstellung wird nun ein Konzept[1] entwickelt, um daraus eine Lösung zu erarbeiten.

2.1 Bestimmung der wesentlichen Anforderungen

Bevor das Konzept erstellt wird, sollen zunächst jedoch die Anforderungen daran dokumentiert werden.

- **Sicherheit**: Die wichtigste Eigenschaft soll die Sicherheit des Verfahrens sein. Die folgenden Punkte sollen daher möglichst wenig Einfluss auf die Sicherheit haben.
- **Einmaligkeit der Einmalpasswörter**: Da jeder Arbeitsplatz generell als unsicher angesehen werden muss, darf jedes Einmalpasswort tatsächlich nur *einmal* verwendet werden, dies muss durch das Verfahren sichergestellt werden.
- **Einfachheit der Benutzung**: Der Benutzerkreis soll möglichst groß sein. Dazu ist es erforderlich, dass das System einfach zu bedienen ist. Auch für Administratoren und Programmierer soll der Aufwand möglichst gering sein.

[1] Dieser Abschnitt setzt Kenntnisse der Informationstheorie voraus, die bei Bedarf in Anhang A.1 auf Seite 18 erläutert werden.

- **Offenheit**: Das gesamte System soll vollständig offengelegt und dokumentiert sein. Die Sicherheit soll im Verfahren liegen, nicht in der Geheimhaltung. Dies ist ein wichtiger Grundsatz, der nicht sofort verständlich ist. In der Fachliteratur wird dieser Grundsatz als *Kerckhoffs' Maxime* bezeichnet (vgl. [Sch96, Seite 8]).

2.2 Entscheidungsfindung über wesentliche Parameter

Unter Berücksichtigung der genannten Anforderungen wird nun ein System entwickelt, das den Anforderungen entspricht. Das System soll für den Benutzer so funktionieren, dass sein Mobiltelefon ein Programm enthält, welches ihm das aktuelle Einmalpasswort anzeigt. Das System soll allerdings nicht nach einer bestimmten Anzahl von Verwendungen zurückgesetzt werden müssen, wie es bei SKEY (siehe Abschnitt 1.3.1 auf Seite 5) der Fall ist, denn dies würde der Anforderung an die Einfachheit widersprechen.

2.2.1 Synchronisation

Trotzdem muss eine Art der Synchronisation zwischen dem Programm des Benutzers und dem System der authentifizierenden Stelle erfolgen, damit eindeutig ist, welches Einmalpasswort momentan gültig ist. Es erscheint sinnvoll, das Einmalpasswort aus der Tageszeit abzuleiten, deshalb wird die *Systemzeit*[2] zur Synchronisation gewählt.

So kann ein Algorithmus entwickelt werden, der jede mögliche Uhrzeit auf ein Einmalpasswort abbildet. Wenn die Uhren von Mobiltelefon und der authentifizierenden Stelle synchron sind, wird von beiden das gleiche Einmalpasswort berechnet und eine Authentifizierung kann durch einen Vergleich erfolgen.

Damit sich das Einmalpasswort bei einer Zeitauflösung von einer Sekunde nicht jede Sekunde ändert, wird die Zeit nur minutengenau aufgelöst. So hätte der Benutzer zwischen 0 und 59 Sekunden Zeit, das Einmalpasswort einzugeben, abhängig von den verbleibenden Sekunden innerhalb der aktuellen Minute. Weil die Eingabedauer dadurch immernoch zu kurz sein kann, sollen die Einmalpasswörter der letzen und der nächsten Minute auch gültig sein. Zu jedem Zeitpunkt gibt es also drei mögliche Einmalpasswörter.

2.2.2 Secret

Die Systemzeit darf nicht die einzige Eingabe für den Algorithmus sein, denn sonst könnte eine Authentisierung durch jede Person erfolgen, die den Algorithmus und die Systemzeit kennt. Deshalb soll der Algorithmus das Einmalpasswort nicht nur aus der Systemzeit, sondern zusätzlich aus einer geheimen Zeichenfolge ableiten, dem sogenannten *Secret*.

[2] das sind die vergangenen Sekunden seit dem 1. Januar 1970 0:00 in der Zeitzone 0 (Greenwich-Zeit)

Es darf nur dem Programm im Mobiltelefon und der authentifizierenden Stelle bekannt sein und sollte möglichst lang gewählt sein, damit ein Erraten durch Ausprobieren aller möglichen Zeichenfolgen ausgeschlossen werden kann. Allerdings darf es auch nicht zu lang sein, da der Benutzer es der authentifizierenden Stelle einmalig mitteilen muss. Nachdem beide das gleiche Secret gespeichert haben, sind sie synchronisiert und der Benutzer darf aus Sicherheitsgründen keine Möglichkeit mehr haben, das Secret nachträglich einzusehen. Abbildung 3 zeigt das grundlegende Prinzip.

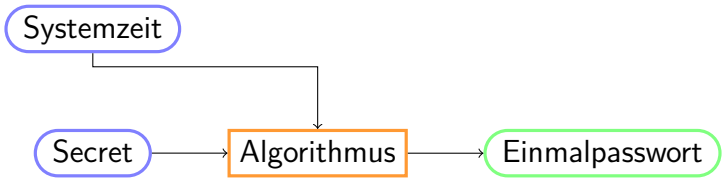


Abbildung 3: Grundlegendes Prinzip des Systems

Eine Länge von 132 Bits für das Secret erscheint als guter Kompromiss, denn so würde ein Supercomputer, der theoretisch 1.000.000.000 Secrets pro Sekunde ausprobieren kann, $\frac{2^{132}\,\text{Secrets}}{10^9 \cdot 60 \cdot 60 \cdot 24 \cdot 365\,\text{Sekunden}} \approx 172 \cdot 10^{21}$ Jahre benötigen, um das Secret zu erraten und dadurch gültige Einmalpasswörter erzeugen zu können.

2.2.3 Alphabet

Weil das Secret bei der Registrierung einmalig abgetippt werden muss, ist es wichtig, das Alphabet der gültigen Zeichen genau festzulegen. Die Zeichen a bis z sowie A bis Z und 0 bis 9 bilden mit einer Anzahl von 62 Zeichen eine gute Grundlage. Da die Zeichen 0 und O sowie l und I auf einigen LC-Displays nicht zu unterscheiden sind, werden O und l aus dem Alphabet entnommen und dafür die vier Zeichen `+`, `-`, `*` und `/` hinzugefügt. Diese Sonderzeichen wurden gewählt, weil sie auch mit internationalen Tastaturlayouts einfach einzugeben sind, da sie auf dem Nummernblock enthalten sind. Das endgültige Alphabet ist in Abbildung 4 zu sehen. Es besteht aus 64 Zeichen.

$$\Sigma = \{\texttt{ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz0123456789+-*/}\}$$

Abbildung 4: Alphabet des Systems

Bei diesem Alphabet trägt jedes Zeichen $\log_2(64) = 6\,\text{bit}$ Informationen, es hat also einen Informationsgehalt von 6 Sh. Die Länge des Secrets und des Einmalpassworts müssen Vielfache davon sein, damit die Zeichen gleichverteilt sind. Zur Eingabe des Secrets sind mit diesem Alphabet $\frac{132\,\text{bit}}{6\,\text{Sh}} = 22$ Zeichen notwendig. Da es sich um eine einmalige Eingabe handelt, ist dieser Wert akzeptabel.

2.2.4 Einmalpasswort

Das Einmalpasswort soll aus einer Zeichenkette bestehen, die der Benutzer von seinem Mobiltelefon abtippen kann. Die wesentliche Anforderung an das Einmalpasswort stellt die Sicherheit. Anders als beim Secret kann die Länge aber kürzer sein, denn im Gegensatz dazu ist das Einmalpasswort nur wenige Minuten gültig. Abbildung 9 auf Seite 20 verdeutlicht den Zusammenhang zwischen der Anzahl der Zeichen im Alphabet und der benötigten Zeichen zur Darstellung einer Information. Das gewählte, in der Abbildung grüne Alphabet kann mit 10 Zeichen noch 60 Bits darstellen, dies erscheint als Kompromiss zwischen Sicherheit und Einfachheit sinnvoll.

Es ergeben sich also 2^{60} (das sind ungefähr 10^{18}) mögliche Einmalpasswörter. Um in den drei Minuten, die ein Einmalpasswort gültig ist, alle Einmalpasswörter auszuprobieren, würde ein Supercomputer benötigt, der pro Sekunde $\frac{2^{60}\,\text{Kombinationen}}{180\,\text{s}} \approx 6{,}4 \cdot 10^{15}$ Einmalpasswörter probieren könnte. Dies ist schon wegen Verzögerungen im Netzwerk oder bei der manuellen Eingabe unrealistisch.

2.2.5 Zusammenfassung der Parameter

Parameter	**Beschreibung**	**Wert**
Zeichen im Alphabet	Anzahl der Zeichen, die dargestellt werden können	64
Bits pro Zeichen	Anzahl Bits, die ein Zeichen darstellt	6
Bits im OTP	Anzahl Bits, die ein Einmalpasswort enthält	60
Anzahl Zeichen OTP	Anzahl Zeichen, die ein Einmalpasswort enthält	10
Bits im Secret	Anzahl Bits, die ein Secret enthält	132
Anzahl Zeichen Secret	Anzahl Zeichen, die ein Secret enthält	22

Tabelle 1: Wesentliche Parameter des Systems

Tabelle 1 zeigt die wesentlichen Parameter in einer Zusammenfassung.

2.3 Beschreibung des theoretischen Systems

Der Algorithmus zur Generierung eines Einmalpassworts erhält als Eingaben die Systemzeit und das Secret des Benutzers und bildet sie als Ausgabe auf ein Einmalpasswort ab. Das Alphabet der gültigen Zeichen sowie die Länge des Secrets und des Einmalpassworts sind Konstanten.

Bei der Abbildung ist es sehr wichtig, dass sie *surjektiv*[3] und *chaotisch*[4] ist. Das surjektive Verhalten ist wichtig, damit die gesamten 60 Bits auch tatsächlich genutzt werden, also

[3] bedeutet, dass jede mögliche Ausgabe tatsächlich vorkommen kann

[4] bedeutet, dass eine minimale Änderung der Eingabe durchschnittlich die Hälfte der Ausgabe verändert

jedes theoretisch mögliche Einmalpasswort auch tatsächlich vorkommen kann. Dadurch ist die Wahrscheinlichkeit für die Einmalpasswörter gleichverteilt, was Angriffe erschwert. Das chaotische Verhalten ist wichtig, weil sich zwischen zwei gültigen Einmalpasswörtern nur die Systemzeit ändert, im schlimmsten Fall nur um ein einzelnes Bit. Wenn die Abbildung chaotisch ist, sehen beide Einmalpasswörter trotz des geringen Unterschieds der Eingabe sehr unterschiedlich aus. Dass sich zwischen zwei Einmalpasswörtern in der Eingabe nur ein Bit ändert, nützt einem Angreifer nichts, da er die 132 Bits des Secrets nicht kennt.

Hashfunktionen Es gibt eine Gruppe von Abbildungen, welche diese Eigenschaften erfüllen, die sogenannten Hashfunktionen. Sie werden in der Regel verwendet, um eine beliebige Eingabe auf eine möglichst eindeutige Ausgabe konstanter Länge abzubilden. Es ist möglich, aber unwahrscheinlich, dass zwei unterschiedliche Eingaben eine identische Ausgabe liefern[Sch96, Seiten 35 f.]. Für die Anwendung in diesem System sollte die Hashfunktion nicht umkehrbar sein. Es soll also unmöglich sein, aus der Ausgabe Eigenschaften über die Eingabe treffen zu können. Diese Hashfunktionen werden Einweg-Hashfunktionen genannt.

Die sogenannten SHA-2-Funktionen sind die aktuell vom NIST, dem amerikanischen Institut für Standards, empfohlenen Einweg-Hashfunktionen[NIS09], weshalb sie hier verwendet werden. SHA-2-Funktionen bilden eine beliebige Eingabe auf 224, 256, 384 oder 512 Bits ab. Eine Größe von 256 Bits ist ausreichend, um daraus ein Einmalpasswort mit 60 Bit zu erzeugen, weshalb sie für dieses System ausgewählt wird. Die genaue Beschreibung von SHA-256 findet sich in [EH06].

Aufbereitung der Eingabe Das Secret liegt als Zeichenkette mit 22 Zeichen vor. Die Systemzeit liegt als Zahl vor, welche die vergangen Sekunden seit dem 1. Januar 1970 0:00 UTC (Universal Coordinated Time, also Zeitzone GMT) darstellt. Diese Zahl wird durch 60 geteilt, um nur die Minuten zu berücksichtigen, damit nicht jede Sekunde eine neue Eingabe entsteht (siehe Abschnitt 2.2.1 auf Seite 7). Die Minuten werden in eine Zeichenkette umgewandelt und an das Secret gehängt, sodass eine neue Zeichenkette entsteht.

Abbildung Die entstandene Zeichenkette wird als Eingabe für die SHA-256-Funktion genutzt, die daraus einen Wert mit 256 Bits Länge erzeugt. Aus diesem Wert soll das Einmalpasswort abgeleitet werden. Dann wäre es allerdings möglich, aus einem erspähten Einmalpasswort Informationen über den Hashwert des Secrets zu erhalten. Wenn zusätzlich noch die Systemzeit bekannt ist, wäre es theoretisch möglich, dass ein Angreifer viele Kombinationen aus Einmalpasswort und Systemzeit sammelt, um damit durch Ausprobieren aller 2^{132} möglichen Secrets festzustellen, welches der Secrets das korrekte ist.

Dieser Angriff ist nur theoretisch möglich, da es praktisch unmöglich ist, 2^{132} Secrets auszuprobieren. Allerdings soll das System auch theoretischen Angriffen standhalten, deshalb wird der berechnete Hashwert einfach nochmal der SHA-256-Funktion übergeben. Dadurch kann der Angreifer nur noch auf den Hash *einer* Kombination schließen, nicht aber die Systemzeit zur Hilfe nehmen, da sie in dem zweiten Hash nicht mehr direkt eingeflossen ist.

Transformation Damit eine lesbare und übersichtliche Ausgabe entsteht, werden die ersten 60 Bits des Hashwerts mit dem Alphabet auf eine Zeichenkette abgebildet, die als Eimalpasswort dient. Dazu werden die 60 Bits in 10 Gruppen mit je 6 Bit zerlegt, da jeder Buchstabe 6 Bit Informationen enthält (siehe Abschnitt 2.2.3 auf Seite 8). Die einzelnen 6-Bit-Werte geben jeweils den Index des Zeichens im Alphabet an, das in der Ausgabe benutzt wird.

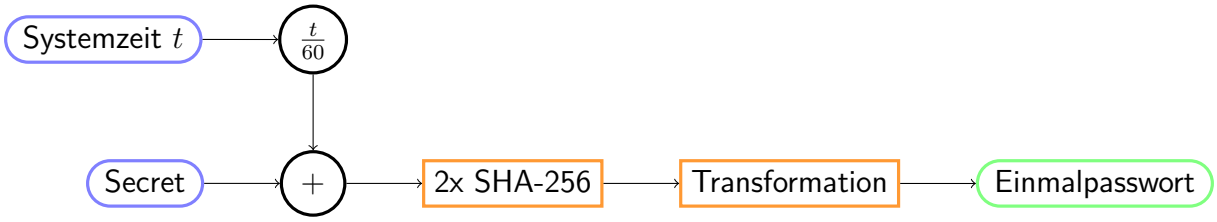


Abbildung 5: Algorithmus zur Erzeugung eines Einmalpassworts

Abbildung 5 zeigt den Algorithmus, Anhang A.2 auf Seite 19 zeigt ein vollständiges Beispiel für die Erzeugung eines Einmalpassworts.

2.4 Entwickeln einer Lösung

Nachdem der Algorithmus vollständig beschrieben ist, soll dieser nun in verschiedenen Formen implementiert werden.

2.4.1 Überprüfung der Einmalpasswörter

Um die Einmalpasswörter zu überprüfen, wird der Algorithmus zur Erzeugung eines Einmalpassworts angewendet. Die authentifizierende Stelle kennt das Secret des Benutzers und kann deshalb die gleichen Einmalpasswörter generieren und sie mit der jeweiligen Eingabe des Benutzers vergleichen.

Damit jedes Einmalpasswort tatsächlich nur einmal gültig ist, speichert die authentifizierende Stelle den Zeitpunkt, an dem die letzte erfolgreiche Anmeldung erfolgte. So kann sie sicherstellen, dass eine erneute Anmeldung erst in der Minute darauf erfolgen darf, wodurch die Anforderung an die Einmaligkeit erfüllt wird. sein darf.

2.4.2 Anzeige der Einmalpasswörter

Für die Implementierung wird ein Mobiltelefon als Ausgabegerät für die Einmalpasswörter gewählt, da in der Regel jeder Mitarbeiter im Besitz eines Mobiltelefons ist. Das Telefon zeigt über ein MIDlet (ein Java-Programm, das auf einem Mobiltelefon ausgeführt wird) das aktuelle Einmalpasswort an.

Damit die Anforderung an die Sicherheit erfüllt bleibt, fragt das Programm beim Starten nach einem konventionellen Passwort, mit dem die Secrets im Telefon verschlüsselt werden. Wenn ein ungültiges Passwort beim Starten eingegeben wird, entschlüsselt das Programm die Secrets falsch und wird deshalb ungültige Einmalasswörter erzeugen. Dadurch kann das Startpasswort nicht erraten oder berechnet werden.

2.5 Zusammenfassung

Die theoretische Beschreibung ist nun vollständig. Der Benutzer kann sich mithilfe seines Mobiltelefons an Systemen anmelden und benötigt zusätzlich ein konventionelles Passwort, um das Programm auf dem Mobiltelefon zu starten. Damit sind die gestellten Grundsätze an die Sicherheit, etwas zu *wissen* und etwas zu *besitzen*, beide erfüllt.

2.5.1 Leistungsfähigkeit

Die Einmalpasswörter haben eine Stärke von 60 Bits. Dies ist ausreichend, denn jedes Einmalpasswort ist nur 180 Sekunden lang gültig. Da die SHA-256-Funktion surjektiv ist, werden tatsächlich alle 2^{60} Einmalpasswörter benutzt. Dass ein Einmalpasswort für denselben Benutzer zweimal vorkommt, ist unrealistisch und wäre trotzdem nicht tragisch.

Um das System anzugreifen, müsste also das Secret ermittelt werden. Das Secret hat allerdings eine Stärke von 132 Bits, weshalb es gänzlich unrealistisch ist, alle möglichen Secrets zu prüfen.

3 Durchführung

Die Implementierungen erfolgen in Ruby, C, Java und PHP. Deshalb ist es sinnvoll, vor der Programmierung ein UML-Diagramm für die Klasse zu erstellen. Es erscheint angemessen, die Klasse gemäß dem *Utility-Pattern* zu erstellen, da sie keinen internen Zustand hat. Die Klasse wird also nur statische Methoden enthalten (vgl. [Kru05, Seite 185]). Der Name der Klasse lautet ModgudAuth. Modgud ist in der germanischen Mythologie eine Riesin, welche die Brücke zum Totenreich Hel bewacht und jeden Ankömmling nach seinem Namen fragt. Abbildung 6 auf der nächsten Seite zeigt das UML-Diagramm der Klasse. Unterstrichene Elemente sind statisch, Elemente mit einem Pluszeichen `public`

und Elemente mit einem Minuszeichen `private`. Tabelle 2 erläutert die Funktionen der einzelnen Attribute und Methoden.

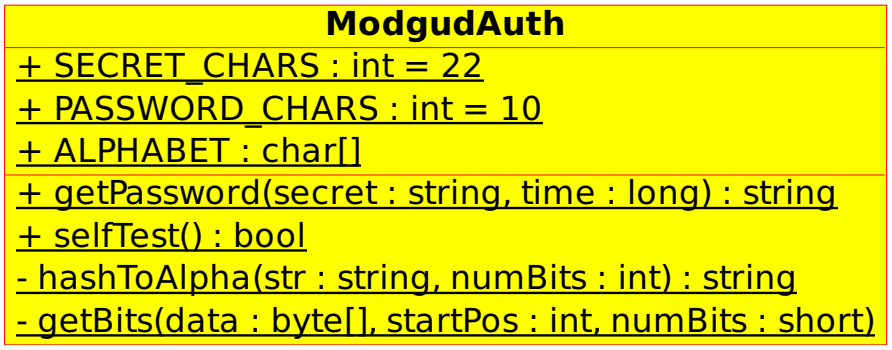


Abbildung 6: ModgudAuth-Klasse als UML-Diagramm

Diese Klasse dient nur der Erzeugung von Einmalpasswörtern. Die Überprüfung eines Einmalpassworts gehört nicht dazu, weil diese Funktionalität nicht in jeder Anwendung nötig ist; das Mobiltelefon muss beispielsweise nur Eimalpasswörter erzeugen, aber niemals prüfen. Außerdem ist diese Funktionalität stark implementationsabhängig. Eine Webanwendung wird das zuletzt genutzte Einmalpasswort in einer Datenbank speichern, ein Systemprogramm hingegen in einer einfachen Datei.

Name	**Funktion**
SECRET_CHARS	Anzahl der Zeichen in einem Secret
PASSWORD_CHARS	Anzahl der Zeichen in einem Einmalasswort
ALPHABET	Alphabet des Systems als Zeichenkette
getPassword	gibt das Einmalpasswort für eine bestimmte Zeit zurück
selfTest	prüft, ob die Implementierung gültig ist
hashToAlpha	hasht eine Zeichenkette und stellt sie mit dem Alphabet dar
getBits	extrahiert eine Anzahl von Bits aus einem Array

Tabelle 2: Attribute und Methoden der Klasse ModgudAuth

Prototyp Zuerst wurde ein Prototyp entwickelt, um das Konzept zu testen, bevor es in den verschiedenen Programmiersprachen implementiert wurde. Dieser Prototyp wurde in der Programmiersprache *Ruby* als *Unit-Test* entwickelt. Folgende Eigenschaften wurden getestet:

- jedes Zeichen des Alphabets wird tatsächlich genutzt
- unter 100.000 fortlaufenden Einmalpasswörtern befindet sich kein Duplikat
- die Häufigkeitsverteilung der Zeichen in den Einmalpasswörtern ist in 100.000 Proben gleichverteilt, die mittlere Abweichung ist geringer als 0,5%

Alle Tests waren erfolgreich, das Konzept wurde deshalb wie geplant umgesetzt.

3.1 Implementierung der Klasse ModgudAuth

Die beschriebene Klasse wurde in den Programmiersprachen C, Ruby, Java und PHP implementiert, um mit verschiedenen Plattformen Einmalpasswörter erzeugen zu können.

3.1.1 Anwendungen

Mit der programmierten Klasse wurden zwei Anwendungen für authentisierende Benutzer realisiert. Es wurde ein Java-Programm für Mobiltelefone entwickelt, das der Verwaltung von verschiedenen authentifizierenden Stellen dient. Das Programm erzeugt für jede Stelle ein zufälliges Secret, das der Benutzer über einen sicheren Kanal der authentifizierenden Stelle mitteilt. Danach hat der Benutzer keine Möglichkeit mehr, das Secret auszulesen. Für jede eingetragene Stelle wird das jeweils gültige Einmalpasswort sowie die verbleibende Eingabezeit angezeigt.

Weiterhin wurde eine Schaltung mit einem MikroController und einem LC-Display entwickelt. Das Programm des Controllers kann über einen PC konfiguriert werden. Es ist möglich, das Secret einzustellen sowie die Zeit zu synchronisieren. Danach zeigt das Display fortlaufend die gültigen Einmalpasswörter und die verbleibende Eingabezeit an.

3.2 Implementierung der Funktion zur Authentifizierung

Die Klasse ModgudAuth bildet auch die Basis zur Überprüfung eines Einmalpassworts, da die Prüfung durch einen Vergleich mit einem von der Klasse erzeugten Einmalpasswort stattfinden kann.

3.2.1 Authentifizieren eines Benutzers

Um einen Benutzer zu authentifizieren, wird das eingegebene Einmalpasswort mit den Einmalpasswörtern der letzten, der aktuellen und der folgenden Minute verglichen.

Sofern es eine Übereinstimmung gibt, wird geprüft, ob die aktuelle Systemzeit in Minuten größer als die des letzten Anmeldezeitpunkts in Minuten ist. Falls dies nicht zutrifft, wurde ein Einmalpasswort zweimal verwendet und es erfolgt keine Authentifizierung. Im anderen Fall hat der Benutzer ein gültiges Einmalpasswort eingegeben und wird authentifiziert. Der Anmeldezeitpunkt wird gespeichert, wodurch das Einmalpasswort der verwendeten Minute sowie aller Minuten davor ungültig wird.

Anstelle des Zeitpunkts hätte auch das eigentliche Einmalpasswort gespeichert werden können, allerdings besteht eine geringe Wahrscheinlichkeit, dass ein Einmalpasswort sich zu einem späteren Zeitpunkt wiederholt.

3.2.2 Anwendungen

Mit der programmierten Funktion wurden drei Anwendungen für authentifizierende Stellen entwickelt. Es wurde eine Webseite zum Testen programmiert, bei der Benutzer sich registrieren und danach authentisieren können. Zusätzlich zum konventionellen Passwort besteht für die Benutzer die Möglichkeit, ein Secret zu hinterlegen und sich danach mit Einmalpasswörtern zu authentisieren. Diese Webseite ist unter der Adresse `http://modgud-test.solhost.org/` erreichbar.

Weiterhin wurde eine PAM-Bibliothek für UNIX-Systeme entwickelt, die es ermöglicht, sich mit Einmalpasswörtern am Betriebssystem anzumelden. Den Benutzern steht ein Befehl zur Verfügung, mit dem sie ein Secret einstellen oder löschen können.

Zuletzt wurde eine GINA-Bibliothek für Windows-Systeme entwickelt, die das Gleiche für Microsoft Windows ermöglicht.

3.3 Besonderheiten bei der Programmierung

Bei einigen Programmiersprachen sind Besonderheiten aufgetreten, die nicht vorhergesehen waren.

Java-MIDlet Beim Test des Programms auf Mobiltelefonen fiel auf, dass nicht jedes Mobiltelefon die nötigen Methoden für die Handhabung von Zeitzonen bereitstellt. Deshalb wurde eine Möglichkeit entwickelt, die Abweichung zwischen der Systemzeit des Mobiltelefons und der UTC-Zeit manuell zu ermitteln, indem der Benutzer beim ersten Start die aktuelle UTC-Zeit eingibt.

PHP Bei der Implementierung in PHP fiel auf, dass sie als einzige der gewählten Programmiersprachen nicht streng typisiert ist. Da dies für die Erzeugung der Einmalpasswörter aber sehr wichtig ist, mussten die Variablen teils manuell in die richtigen Typen konvertiert werden, beispielsweise bei der Teilung zweier Integer, die in anderen Sprachen immer zu einem Integer führt, in PHP allerdings zu einer Float, die dann manuell in einen Integer konvertiert werden muss.

C Die Programmierung der Klasse und der Anwendungen in C war die größte Herausforderung. Einerseits musste darauf geachtet werden, dass die erstellte Klasse auf verschiedenen Plattformen (UNIX, Windows) als auch auf verschiedenen CPU-Architekturen (PC, AVR-MikroController) lauffähig ist. Andererseits musste beim Programm zum Einstellen des Secrets bedacht werden, dass das Programm Zugriff auf die Datei mit den gespeicherten Secrets benötigt, damit der Benutzer über das Programm sein Secret eintragen

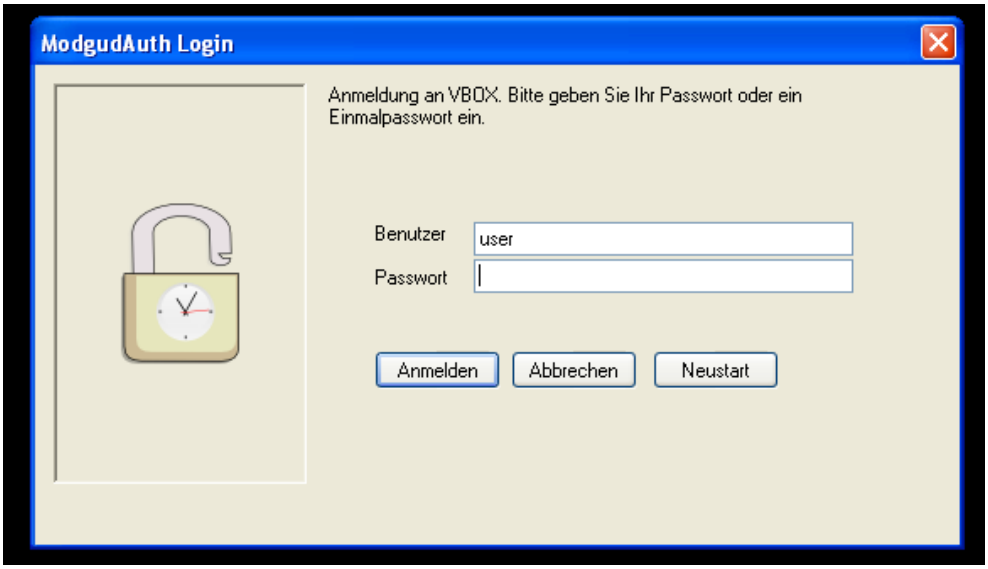
kann. Der Benutzer selbst darf auf keinen Fall Leserechte für diese Datei haben. Da ein Programm aber üblicherweise mit den Rechten des Benutzers läuft, musste eine andere Möglichkeit gewählt werden, dem Programm Zugriff auf die Datei zu geben. Dies wurde über das SUID-Bit UNIX' realisiert, das dafür sorgt, dass ein Programm mit den Rechten des Eigentümers ausgeführt wird, anstatt mit den Rechten des Anwenders. Die Tiefen der Systemprogrammierung wurden dem Buch [Her04] entnommen.

4 Vorstellung des Systems

Dieser Abschnitt zeigt einige Eindrücke des fertigen Systems.

(a) Java-MIDlet (b) MikroController

Abbildung 7: Implementierungen für Benutzer



(a) GINA-Modul für Microsoft Windows

Abbildung 8: Implementierung für authentifizierende Stellen

5 Fazit

5.1 Vergleich mit bestehenden Systemen

Der administrative Aufwand bei ModgudAuth ist gering, denn im Gegensatz zu anderen Systemen muss das Mobiltelefon nur einmalig mit der authentifizierenden Stelle synchronisiert werden, indem Secret und Zeit abgeglichen werden. Im Vergleich zu kommerziellen Systemen wie SecurID wird dieses System kostenlos zur Verfügung gestellt und erlaubt es jedem Nutzer, die Sicherheit zu überprüfen, da der Quelltext frei ist. Dies ist ein weiterer Vorteil gegenüber kommerziellen Systemen.

Tabelle 3 vergleicht die bestehenden Systeme mit dem eigenen unter Berücksichtigung der gestellten Anforderungen.

System	**Offenheit**	**Einfachheit**	**Sicherheit**	**Kosten**
SKEY	Ja	Nein	Ja	Frei
RFC2289	Ja	Nein	Ja	Frei
SecurID	Ja (seit 2003)	Ja	Ja	Hoch
ModgudAuth	Ja	Ja	Ja	Frei

Tabelle 3: Vergleich bestehender Systeme mit dem eigenen

5.2 Selbstreflexion

Die Entwicklung des Systems barg einige Schwierigkeiten. Die Programmierung der Methode `getBits` (siehe Anhang A.6 auf Seite 25) stellte eine Herausforderung an die Programmierfähigkeit dar, weil es für den Autor schwierig war, von einem Array in die einzelnen Bits und Bytes umzudenken, die darin enthalten sind, um so an einzelne Bitgruppen zu gelangen.

Anderseits war es oft sehr mühsam, bei einer Änderung des theoretischen Systems alle Implementierungen wieder und wieder anzupassen, weil der Autor dachte, dass dies die letzte Änderung gewesen sei. Es wurde zwar im Vorwege ein Prototyp entwickelt, um das gesamte System vor der Übersetzung in die anderen Programmiersprachen zu testen, allerdings ergaben sich die meisten Änderungen erst *bei* den verschiedenen Übersetzungen.

Insgesamt ist der Autor aber sehr zufrieden mit seiner Arbeit und setzt das entwickelte System auch für sich selbst ein, um sich auf Reisen mit einem VPN zu verbinden und dabei sicher zu authentisieren.

A Anhänge

A.1 Informationstheoretische Grundlagen

Als *Bit* wird in der Informationstheorie die kleinstmögliche Einheit für eine Information bezeichnet. Es unterscheidet nur zwischen zwei Zuständen. Ein Beispiel für so einen Zustand ist ein einzelner Lichtschalter, er hat nur die Zustände *an* und *aus*. Kommt zu diesem Lichtschalter ein weiterer, können beide insgesamt 2 Bits an Informationen tragen, es gibt insgesamt 4 Zustände. Kommt noch ein weiterer hinzu, sind es insgesamt 3 Bits. Allerdings sind dies nicht 6 Zustände, sondern 8. Tabelle 4(a) zeigt, warum. Es gibt insgesamt also $2 \cdot 2 \cdot 2 = 2^3$ Zustände, mit jedem Schalter verdoppelt sich die Anzahl der Möglichkeiten.

Tabelle 4: Zusammenhänge zwischen Bits und Zuständen

(a) Mögliche Zustände bei 3 Schaltern

S_1	S_2	S_3
aus	aus	aus
aus	aus	ein
aus	ein	aus
aus	ein	ein
ein	aus	aus
ein	aus	ein
ein	ein	aus
ein	ein	ein

(b) Darstellung mit einem Alphabet aus 26 Zeichen

Zeichen	S_1	S_2	S_3	S_4	S_5
A	aus	aus	aus	aus	aus
B	aus	aus	aus	aus	an
C	aus	aus	aus	an	aus
D	aus	aus	aus	an	an
E	aus	aus	an	aus	aus
F	aus	aus	an	aus	an
...	...	...	...	...	...
Z	an	an	an	an	an

Ein anderes Beispiel: Im lateinischen Alphabet gibt es 26 Buchstaben. Um zu wissen, wie viele Bits benötigt werden, um sie voneinander zu unterscheiden, muss die Gleichung $26 = 2^x$ gelöst werden. Die Umkehrung von $y = a^x$ ist $x = \log_a(y)$. Die Lösung lautet also $\log_2(26) \approx 4{,}7$. Nun gibt es aber nur ganze Bits, deshalb muss das Ergebnis *immer* aufgerundet werden: $\lceil \log_2(26) \rceil = 5$. Es genügen also 5 Bits, um jeden Buchstaben so darzustellen, dass er von anderen unterschieden werden kann. Jedes Zeichen hat einen *Informationsgehalt* von 5 Sh (Einheit Shannon $\Leftrightarrow \frac{bit}{Zeichen}$).

Angenommen, nur eine bestimmte Kombination aus 100 Schaltern schaltet das Licht an. Es ist möglich, diese Information darzustellen, indem 100 mal die Wörter „an“ oder „aus“ aneinandergereiht werden. Allerdings wäre es mühselig, dies zu notieren. Die 100 Bits an Informationen, die in der Kombination enthalten sind (nämlich welcher Schalter welchen Zustand hat), lassen sich auch mit dem gerade entwickelten Alphabet darstellen. Da jedes Zeichen 5 Bit trägt, werden dann nur $\frac{100\,\text{bit}}{5\,\text{Sh}} = 20$ Zeichen dieses Alphabets zur

Darstellung benötigt. Jedes Zeichen stünde dann für eine eindeutige Kombination aus fünfmal an und aus, wie Tabelle 4(b) auf der vorherigen Seite zeigt.

Werden dem Alphabet noch weitere Zeichen hinzugefügt, beispielsweise Kleinbuchstaben und Ziffern (62 Zeichen insgesamt), ergibt sich ein Informationsgehalt von $\lceil \log_2(62) \rceil = 6\,\mathrm{Sh}$. Dann würden nur noch $\lceil \frac{100\,\mathrm{bit}}{6\,\mathrm{Sh}} \rceil = 17$ Zeichen benötigt. Die enthaltene Information ist aber genau die gleiche. In der Informatik werden Alphabete häufig mit dem Zeichen Σ bezeichnet, $|\Sigma|$ ist die Anzahl der Zeichen in einem Alphabet.

A.2 Beispiel für den Algorithmus

Für Donnerstag, den 19. März 2009 11:27:33 Uhr, beträgt die Systemzeit 1237458453 Sekunden. Geteilt durch 60 ergibt dies 20624307 Minuten. Als vorher zufällig bestimmtes Secret wird `xbCcNh-F916uSCrRVENwnj` gewählt. Aus der Eingabe wird also die Zeichenkette `xbCcNh-F916uSCrRVENwnj20624307` gebildet.

Diese Zeichenkette wird von SHA-256 auf den Wert `0289edd6 4a9b55c4 e43f9d00 7f-636d80 45af6e32 6d7df5f2 6fc5dfc1 e87c83d7` abgebildet. Die Darstellung erfolgt in diesem Beispiel zur Übersicht hexadezimal.

Dieser Wert wird nochmals SHA-256 übergeben (*nicht* als hexadezimale Zeichenkette, sondern als tatsächlicher Wert), daraus entsteht der Hash `2382b2fc 484d043a ba06c52d b1209c1e e99b1188 dca2570c 623a6e36 a91fcce5`.

Da jede hexadezimale Ziffer 4 Bits darstellt, lauten die ersten 60 Bits des Hashes folglich `2382b2fc484d043`. In binärer Schreibweise ergibt das den Wert `001000 111000 001010 110010 111111 000100 100001 001101 000001 000011`. Zur Übersicht wurde zwischen 6 Bits jeweils ein Leerzeichen gesetzt, denn dies sind schon die 6er-Gruppen.

In dezimale Werte übertragen ergeben die 10 binären Gruppen die Werte 8, 56, 10, 50, 63, 4, 33, 13, 1, 3. Diese werden nun als Index für das Alphabet genutzt, wobei die Zählung bei 0 beginnt. Es ergibt sich also das Einmalpasswort `I6K0/EiNBD`.

Eine Minute später lautet die Systemzeit 1237458513, also 20624308 Minuten. Als Eingabe ergibt sich demzufolge `xbCcNh-F916uSCrRVENwnj20624308`. SHA-256 bildet dies nach zwei Aufrufen auf `4c5e860a b91a4651 63fb880c a4615c90 336ddda4 7413a5d9 bb0a3e19 2f9b8fde` ab. Die ersten 60 Bit in dezimaler Schreibweise lauten 19, 5, 58, 6, 2, 43, 36, 26, 17, 37. Dies entspricht dem Einmalpasswort `UF8GCtmbSn`. Eine minimale Änderung der Eingabe bewirkte also eine große Änderung der Ausgabe, der Algorithmus funktioniert wie geplant.

A.3 Erläuternde Abbildungen

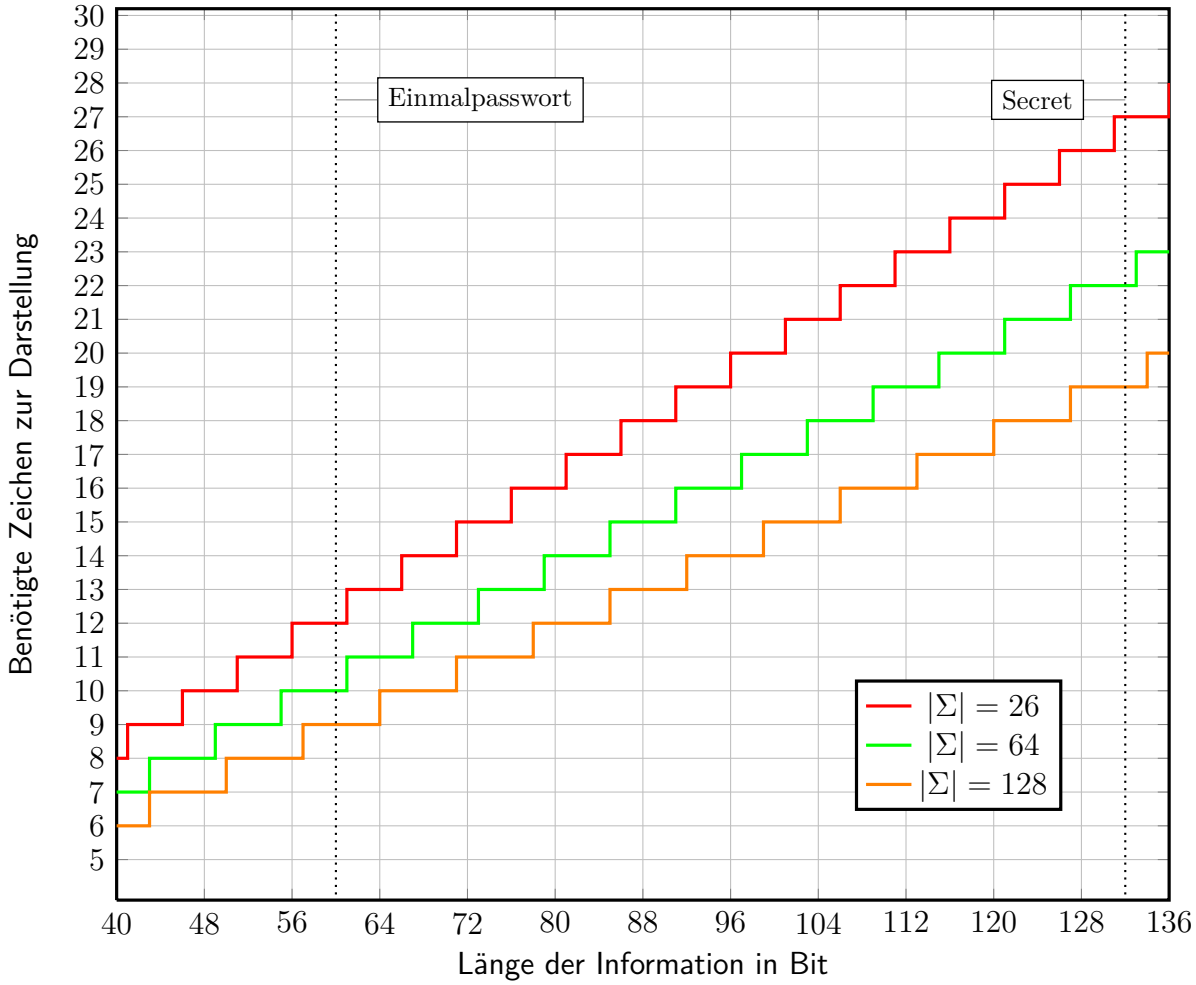


Abbildung 9: Benötigte Zeichen zur Kodierung mit verschiedenen Alphabeten

Diese Abbildung bezieht sich auf Abschnitt 2.2.3 auf Seite 8. Siehe dazu auch Anhang A.1 auf Seite 18. Für das Secret wurden 132 Bits gewählt, für das Einmalpasswort 60 Bits. Die Abbildung zeigt, wieviele Zeichen jeweils in Alphabeten mit verschiedener Zeichenanzahl benötigt würden.

A.4 Anleitung für Benutzer

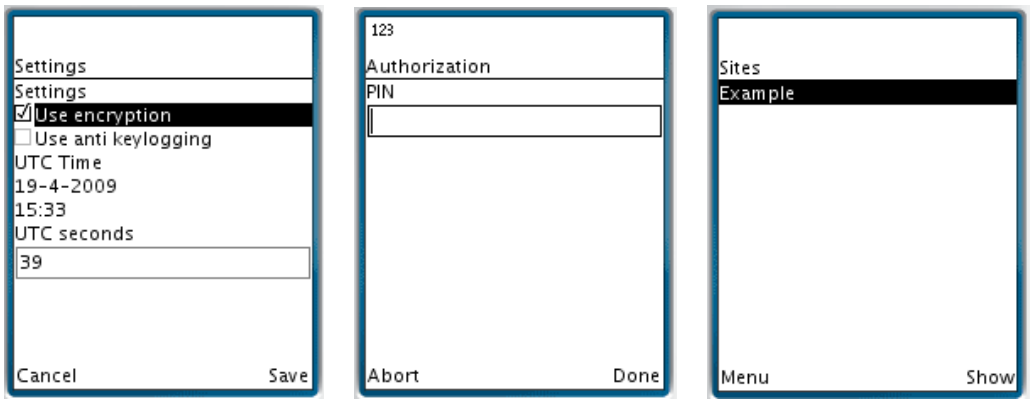
ModgudAuth ist ein System zur sicheren Anmeldung an Diensten. Es wird benutzt, um gegenüber einem Dienst nachzuweisen, dass Sie tatsächlich derjenige sind, für den Sie sich ausgeben.

A.4.1 Installation

ModgudAuth wird als Java-MIDlet geliefert, das Sie auf Ihrem Mobiltelefon installieren können. Kopieren Sie dazu die beiden Dateien `ModgudAuth.jar` und `ModgudAuth.jad` auf Ihr Mobiltelefon. Näheres dazu finden Sie in der Anleitung Ihres Telefons.

A.4.2 Konfiguration

Beim ersten Start erscheint das Konfigurationsmenü (Abbildung 10(a)). Die Einstellungen haben folgende Bedeutung:



(a) Konfiguration (b) PIN (c) Hauptmenü

Abbildung 10: Konfiguration und Hauptmenü

- **Use encryption**: Wenn diese Einstellung aktiviert ist, werden die Informationen im Programm verschlüsselt. Dazu muss beim Starten eine achtstellige PIN eingegeben werden. Es wird empfohlen, diese Einstellung zu aktivieren.
- **Use anti keylogging**: Wenn diese Einstellung aktiviert ist, wird die PIN beim Starten auf eine Weise abgefragt, die es verhindert, dass die Tastendrücke aufgezeichnet werden könnten, um an die PIN zu gelangen. Dazu muss eine zufällige Ziffer zu jeder Ziffer der PIN addiert werden, der Übertrag wird eingegeben. Diese Einstellung sollte nur in kritischen Bereichen verwendet werden.

- **UTC time**: Hiermit haben Sie die Möglichkeit, die Abweichung der Zeit Ihres Telefons von der Weltzeit einzustellen. Zur deutschen Sommerzeit muss diese Zeit zwei Stunden zurückliegen, in der Winterzeit nur eine.
- **UTC seconds**: Hier können Sie die Sekunden eingestellt werden, falls Ihr Telefon bei der Zeiteinstellung keine Sekunden anzeigt.

Hinweis: Die Einstellungen zur Uhrzeit werden nur übernommen, wenn Sie eine der Einstellungen verändert haben. Im Normalfall drücken Sie einfach auf Save. Sie können die Zeit später noch umstellen.

Falls Sie die Verschlüsselung aktiviert haben, werden Sie nun nach der PIN gefragt, die Sie verwenden möchten (Abbildung 10(b) auf der vorherigen Seite). Geben Sie in diesem Feld 8 Ziffern ein. Achtung: Wenn Sie beim nächsten Start des Programms eine falsche PIN eingeben, wird kein Hinweis darüber angezeigt, das Programm erzeugt dann allerdings falsche Einmalpasswörter. Dies ist ein aus Sicherheitsgründen erwünschter Effekt.

A.4.3 Anwendung

Nach der Eingabe der PIN wird das Hauptmenü gezeigt (Abbildung 10(c) auf der vorherigen Seite). Hier sehen Sie eine Übersicht der Seiten, die momentan vom Programm verwaltet werden. Die Seite *Example* kann verwendet werden, um die Funktion zu prüfen. Rufen Sie dazu die Internet-Adresse `http://modgud-test.solhost.org/` in Ihrem Browser auf, auf der Sie weitere Informationen finden. Diese Seite können Sie auch nutzen, um die Uhr im Programm mit der Weltzeit zu synchronisieren.

Wenn Sie die Seite auswählen, werden das aktuelle Einmalpasswort und die verbleibende Zeit zur Eingabe angezeigt (Abbildung 11(a) auf der nächsten Seite).

Hinweis: Die Zeichen l und O (kleines el und großes O) kommen nicht in den Einmalpasswörtern vor. Es handelt sich um die Zeichen I und 0 (großes i und Ziffer 0), die sich auf einigen Telefonen nicht unterscheiden lassen.

A.4.4 Anlegen einer neuen Seite

Um eine neue Seite anzulegen, wählen Sie im Menü (Abbildung 11(b) auf der nächsten Seite) die Option *New Site*. Daraufhin erscheint der Dialog zum Anlegen einer neuen Seite (Abbildung 11(c) auf der nächsten Seite). Das Feld *Secret* enthält ein zufällig erzeugtes Secret, das Sie der Seite mitteilen. Den Namen können Sie frei wählen.

(a) Einmalpasswort (b) Menü (c) Neu

Abbildung 11: Anlegen einer neuen Seite

Achtung: Notieren Sie sich das Secret nirgends. Sie müssen es nur einmalig der Seite mitteilen und benötigen es dann nicht mehr.

A.4.5 Festlegen des Secrets auf UNIX-Rechnern

Um sich auf einem UNIX-Rechner via ModgudAuth anzumelden, müssen Sie zuerst Ihr Secret einstellen. Legen Sie dazu eine neue Seite im Programm an und verwenden Sie auf dem Rechner dann das Programm `modgud-passwd`, um Ihr Secret einzustellen. Geben Sie das Secret als Parameter beim Aufruf an. Danach ist Ihr Secret eingestellt und Sie können sich via ModgudAuth anmelden. Mit `modgud-passwd 0` können Sie die Anmeldung via ModgudAuth deaktivieren. Zur Änderung des Secrets benötigen Sie ihr normales Systempasswort, das beim Starten abgefragt wird.

A.5 Inhalte der DVD

Die beigelegte DVD enthält den vollständigen Quelltext des Systems, diese Dokumentation, einen MIDlet-Emulator zum Testen des MIDlets sowie eine virtuelle Maschine, um das PAM-Modul zu testen. Die virtuelle Maschine ist im VMWare-Format, eine Beilegung des VMWare-Players war aus lizenzrechtlichen Gründen leider nicht erlaubt. Er kann unter der Adresse `http://www.vmware.com` bezogen werden.

Es folgt eine Auflistung der Inhalte der DVD:

- code: enthält den Quellcode des Systems
  - prototype: enthält den Quellcode des Prototyps in der Programmiersprache Ruby. Das Ausführen des Skripts `test_modgud.rb` startet die beschriebenen Tests.
- raw: enthält die Implementierungen in verschiedenen Programmiersprachen
  - C: enthält die Implementierung in C sowieso ein Beispielprogramm
  - Java: enthält die Implementierung in Java sowie ein Beispielprogramm
  - PHP: enthält die Implementierung in PHP sowie ein Beispielskript
  - Ruby: enthält die Implementierung in Ruby sowie ein Beispielskript
- sys: enthält die Implementierungen der Beispielsysteme
  - avr: enthält die Implementierung für MikroController
  - midlet: enthält die Implementierung des Java-MIDlets für Mobiltelefone
  - rails: enthält den Quellcode des Ruby-on-Rails-Projekts, das unter der Adresse `http://modgud-test.solhost.org` veröffentlicht wurde
  - pam: enthält den Quellcode des PAM-Moduls für UNIX-Systeme

Weiterhin gibt es eine Seite im Internet, mit der das System getestet werden kann: `http://modgud-test.solhost.org/`

A.5.1 Virtuelle Maschine

Bei der virtuellen Maschine handelt es sich um ein Linux-System, in welches das entwickelte PAM-Modul integriert ist. Der Benutzername lautet `modgud`, das Passwort ebenso. Allerdings kann zur Anmeldung am Betriebssystem alternativ ein Einmalpasswort benutzt werden, das sich aus dem in Anhang A.2 auf Seite 19 genannten Secret ableitet.

Auf dem Desktop des Systems befindet sich eine Verknüpfung, über die ein Mobiltelefon-Simulator gestartet wird, der das entwickelte MIDlet enthält. Das Secret, welches gültige Einmalpassworte für die Maschine erzeugt, ist dort hinterlegt.

A.6 Ausgewählter Quellcode des Systems

Dieser Anhang zeigt die Funktion, welche einzelne Bits aus einem Array extrahiert.

```
/* extract ,num_bits' from ,data' starting at ,start_pos' */
uint8_t modgud_get_bits(uint8_t *data, size_t data_len,
                        size_t start_pos, uint8_t num_bits)
{
  size_t start_byte = start_pos / 8;
  uint8_t start_bit = start_pos % 8;

  /* get num_bits 1-bits, e.g. 0b111111 for num_bits = 6 */
  uint8_t mask = modgud_get_mask(num_bits);
  uint8_t val;

  if(start_bit + num_bits <= 8) { /* bits are only in one byte */
    if(start_byte >= data_len) { /* out of bounds */
      return -1;
    }
    val = data[start_byte] >> (8 - num_bits - start_bit);
  } else { /* bits span two bytes */
    uint8_t bits1, bits2, val1, val2;

    if(start_byte + 1 >= data_len) { /* out of bounds */
      return -1;
    }
    bits1 = 8 - start_bit;
    bits2 = num_bits - bits1;

    /* get bits from byte 1 */
    val1 = (data[start_byte] & modgud_get_mask(bits1)) << bits2;

    /* get bits from byte 2 */
    val2 = ((data[start_byte + 1]) >> (8 - bits2))
             & modgud_get_mask(bits2);

    /* join bits */
    val = val1 | val2;
  }
  return val & mask;
}
```

B Glossar

Algorithmus	Vorgehensweise, die zur Lösung eines Problems als Abfolge von Schritten beschrieben wird.
Authentifizierung	Überprüfung der Identität eines Benutzers.
Authentisierung	Beweisen der eigenen Identität.
Einmalpasswort	ein Passwort, das nur *einmal* und nur für begrenzte Zeit gültig ist. Im Englischen als OTP (One-Time-Password) bezeichnet.
Hash-Funktion	eine mathematische Funktion, die eine Eingabe auf einen möglichst eindeutigen Wert abbildet.
Kerckhoffs' Maxime	Grundsatz, dass die Sicherheit eines Systems nicht von dessen Geheimhaltung abhängen darf.
Keylogger	eine Software oder Hardware, die Tastatureingaben speichert, damit diese später ausgewertet werden können, um an Passwörter des Benutzers zu gelangen.
konventionelles Passwort	ein Passwort, das sich ein Benutzer merkt und mehrmals verwendet.
MIDlet	Ein Java-Programm, das auf einem Mobiltelefon ausgeführt werden kann.
Secret	eine geheime Zeichenkette, aus der die Einmalpasswörter gebildet werden..
SecurID	ein System für Einmalpasswörter von der Firma RSA. Benutzt kleine Geräte, die den Benutzern das aktuelle Einmalpasswort anzeigen.
SKEY	ein System für Einmalpasswörter, bei dem jeder Benutzer eine Liste mit 100 Einmalpasswörtern erhält.
Systemzeit	Die aktuelle Uhrzeit im Format Sekunden seit dem 1. Januar 1970 nach UTC (Weltzeit).
TAN-Liste	bei Online-Banking verwendete Liste von Einmalpasswörtern.

C Literatur

[Bru05] Bruns, Martina: Bedrohung durch Keylogger unterschätzt. In: *Heise-Online* (2005). `http://www.heise.de/security/news/meldung/66308` (Einmal erwähnt in Abschnitt 1.2)

[EH06] Eastlake, D. ; Hansen, T.: US Secure Hash Algorithms (SHA and HMAC-SHA). In: *Request for Comments* (2006), Nr. 4634. `http://tools.ietf.org/html/rfc4634` (Einmal erwähnt in Abschnitt 2.3)

[Her04] Herold, Helmut: *Linux/Unix Systemprogrammierung*. 3. Auflage. Addison-Wesley, 2004. – ISBN 3827321603 (Einmal erwähnt in Abschnitt 3.3)

[HMNS98] Haller, N. ; Metz, C. ; Nesser, P. ; Straw, M.: A One-Time Password System. In: *Request for Comments* (1998), Nr. 2289. `http://tools.ietf.org/html/rfc2289` (2 Erwähnungen in den Abschnitten 1.3.2 und D)

[Kre08a] Krempl, Stefan: Bundestag verabschiedet BKA-Gesetz mit heimlichen Online-Durchsuchungen. In: *Heise-Online* (2008). `http://www.heise.de/newsticker/meldung/118812` (Einmal erwähnt in Abschnitt 1.1)

[Kre08b] Krempl, Stefan: "Bundestrojaner"heißt jetzt angeblich "Remote Forensic Software". In: *Heise-Online* (2008). `http://www.heise.de/newsticker/meldung/93807` (Einmal erwähnt in Abschnitt 1.1)

[Kru05] Krueger, Guido: *Handbuch der Java-Programmierung*. 4. Auflage. Addison-Wesley, 2005. – ISBN 3827322014 (Einmal erwähnt in Abschnitt 3)

[NIS09] NIST: Secure Hashing - Approved Algorithms. (2009). `http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html` (Einmal erwähnt in Abschnitt 2.3)

[RS09] RSA-Security: *Securing Your Future with Two-Factor Authentication*. `http://www.rsa.com/node.aspx?id=1156`. Version: 20.02.2009 (Einmal erwähnt in Abschnitt 1.3.3)

[Sch96] Schneier, Bruce: *Angewandte Kryptographie*. 5. Auflage. Addison-Wesley, 1996. – ISBN 3893198547 (4 Erwähnungen in den Abschnitten 1.3, 1.3.1, 2.1 und 2.3)

[Tan01] Tanenbaum, Andrew S.: *Modern Operating Systems*. 2. Auflage. Pearson Education, 2001. – ISBN 0130926418 (2 Erwähnungen in den Abschnitten 1 und 1.3)

D Über dieses Dokument

Danksagung

Mein Dank gilt Prof. Dr. Till Tantau vom Institut für Theoretische Informatik der Universität zu Lübeck für *TikZ* und *Beamer*, Professor Donald Knuth für TEX und PhD Leslie Lamport für LATEX sowie das nach ihm benannte Lamport-OTP-System, auf dem diese Arbeit und auch andere Systeme[HMNS98] basieren.

Lizenz

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich das vorliegende Dokument ohne fremde Hilfe erstellt habe. Stellen, die aus anderen Arbeiten entnommen wurden, sind als solche gekennzeichnet.

Folke Will